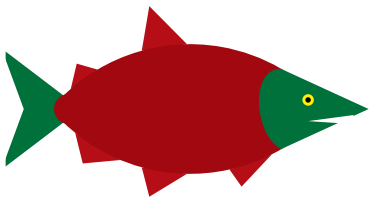


Noch **E**ine **R**aspberry **K**urz**A**nleitung

Consolation for Linux dilettantes

(Linux ohne X mit dem Raspberry Pi)

Alfred H. Gitter

Version vom 29. Januar 2024

Das vorliegende Manuskript ist eine langsame Baustelle und soll erst nach Vollendung der Sagrada Família fertig werden.

Betriebssystem Raspberry Pi OS Lite 32-bit (2023-12-11)

Obwohl hier auf ein Fenstersystem wie X oder Wayland verzichtet wird, ist das meiste auch mit Desktop gut verwendbar.

Informationen und Ratschläge werden `ohne Gewähr` gegeben!

Kritik / Aufmunterung per E-Mail: alfred.gitter@eah-jena.de

Dieses Manuskript ist im Internet verfügbar:

https://www.mt.eah-jena.de/doc/AGitt/raspi.pdf

oder, meistens aktueller: https://ahg88.webnode.com/

Inhaltsverzeichnis

1 Grundlagen

1.1 Vorbemerkungen

1.1.1 Vorwort

Ein Vorwort ist der Preis, den der Leser eines guten Buches gedanklich entrichtet. Ein geiziger oder ungeduldiger Leser kann es natürlich überschlagen und gleich mit der spannenden Technik beginnen. Wer hier jedoch weiterliest, muss sich nun einen geschichtlichen Rückblick gefallen lassen.

In einer Elektronikzeitschrift aus dem Jahre 1979 wurden Mikrocomputer zum Lernen und Basteln angeboten, die den Mikroprozessor Z80 enthielten. Diese Vorgänger des Raspberry Pi wurden bereits mit Tastatur und Bildschirm (Fernseher) betrieben und verfügten über digitale I/O-Ports ähnlich den GPIO des Raspberry Pi. Der Preis für ein System (Platine und Tastatur) mit 2 KiB ROM (für das unveränderbare Betriebssystem) und 2 KiB RAM (Arbeitsspeicher) betrug etwa 1000 DM. Wer wissen will, was *DM* ist und wie viel 1000 DM in heutiger Währungseinheit entsprechen, soll seine Großeltern über die gute alte Zeit befragen.

Gespeichert wurden Programme und Daten auf Audiokassetten in Form von binärem Piepsen. Wer wissen will, was eine *Audiokassette* ist, ...

Es gab noch kein World Wide Web und schnelle Textnachrichten wurden per Telegramm versandt. Wer wissen will, was ein *Telegramm* ist, ...

Ansonsten war es 1979 nicht viel anders als heute. Die künstliche Intelligenz war kurz vor dem Durchbruch und das baldige Ende fossiler Brennstoffe beschlossene Sache.

Im Jahr 1979 sandten uns Raumsonden der NASA (Voyager 1 und 2) schöne Photos vom Jupiter und seinen Monden. Im Jahr 2023 startete die ESA eine Raumsonde (JUICE) zum Jupiter, um ab 2031 hoffentlich schöne Photos vom Jupiter und seinen Monden zu senden.

Gleichfalls im Jahr 1979 begann ich mit meiner Diplomarbeit. Mein damaliger Chef riet mir, mich für Wissenschaft und Technik zu begeistern, denn damit hätte man auch noch Spaß, wenn man mal über 50 ist. Glaubt es nur!

Verändert hat sich allerdings die veröffentlichte deutsche Sprache. Anders als in der zweiten Hälfte des zwanzigsten Jahrhunderts, hat sie im einundzwanzigsten wieder die Aufgabe, eine vereinheitlichte moralische Weltsicht zu verdeutlichen. Damit werden zum Beispiel Spieler als *Gamerinnen und Gamer* bezeichnet. In dieser Bezeichnung fehlt aber noch die Inclusion des und der Diversen. Darum habe ich das betroffene Kapitel *Games* (siehe Seite 158) ins Englische übersetzt und bin der Meinung, dass die Darstellung damit für deutschsprachige Lebensformen lesbarer geworden ist.

1.1.2 Rechenleistung und Betriebssystem (OS)

Rechenleistung

Wir benutzen einen Raspberry Pi. Welches Modell? Egal. Die Programme laufen auch auf einem Raspberry Pi 1 B mit einem Arbeitsspeicher von 512 MiB (für CPU und GPU).

Es spricht aber nichts gegen einen Raspberry Pi 5-Boliden, besonders für Multimedia-Anwendungen oder wenn man mit dem Textsatzsystem LaTeX ein Manuskript schreibt.

Betriebssystem (OS)

Das vorliegende Manuskript verwendet als Betriebssystem *Raspberry Pi OS* der Raspberry Pi Foundation in der Version *Lite* mit 32 Bit. Die Beschränkung auf die 32-Bit-Variante ermöglicht es, das Betriebssystem auch mit alten Modellen des Raspberry zu betreiben. Für neuere Modelle (ab 3B) kann man *Lite* mit 64 Bit installieren. Damit können mehr als vier Gigabyte Arbeitsspeicher verwaltet werden (wenn man soviel hat). Ich vermute, dass es für die 64-Bit-Variante bei neueren Modellen keine bedeutenden Nachteile gibt, habe es aber nicht geprüft.

Raspberry Pi OS wurde früher (bis Mai 2020) Raspbian genannt. Die Abkürzung OS steht im Englischen für operating system (Betriebssystem.)[1] Lite ist die Version ohne einen fensterorientierten display server (Fenstersystem), wie das X Window System, Wayland oder Mir.

Raspberry Pi OS Lite enthält scheinbar keine graphische Benutzeroberfläche (GUI = graphical user interface) und hat keinen „Desktop“. Sichtbar ist zunächst nur ein „Terminal“, auf dem man in Textform über Tastatur und Monitor mit dem Rechner arbeitet, in der Regel ohne Maus. Trotzdem ist es möglich, Bilder oder Videos zu zeigen. Und auch ohne Fenster ist die Lage nicht aussichtslos (pun intended).

Ist es aber sinnvoll, fensterfrei ohne Desktop und Maus zu arbeiten? Die vorliegenden Aufzeichnungen sind einerseits für Dilettanten, die das Abenteuer suchen. Andererseits ist die Kommandozeile eines Terminals oft effizienter und mächtiger als mausabhängige Programme mit graphischem Ballast.[2]

Schön ist: Man muss nicht auf eine graphische Oberfläche verzichten, um Programme im Terminal zu nutzen. Wenn man das Betriebssystem *Raspberry Pi OS with desktop* benutzt, kann man zwischen einer virtuellen Konsole und dem „desktop“ umschalten, siehe Abschnitt 3.2.1 auf Seite 83.

Vieles, was im vorliegenden Manuskript behandelt wird, funktioniert auch in einem Terminal, der Teil einer graphischen Oberfläche ist, zum Beispiel LXTerminal oder XTerm.[3] Allerdings gibt es kleine Unterschiede in der Funktionsweise verschiedener Terminals, aber daran wird man sich schnell gewöhnen.

---

[1] Es gibt allerdings auch englischsprachige Informatiker, die ihr Betriebssystem als BS bezeichnen.

[2] Sind wir damit schon Hacker? Nein. Hacker benutzen Arch Linux.

[3] Im LXTerminal oder XTerm funktioniert zum Beispiel nicht das PDF-Anzeigeprogramm `fbgs`, aber dafür gibt es im „desktop“ andere Programme. Die Bildanzeige mit `fim` funktioniert im LXTerminal oder XTerm, während `fbi` nur mit Administratorrechten geht: `sudo fbi -T 1` *bilddatei*

Das vorliegende Manuskript wurde an Raspberry Pi OS Lite in der Version vom 11. Dezember 2023 angepasst. Es kann aber nicht gewährleistet werden, dass die Aktualisierung vollständig ist! Grundlage ist Debian 12 (bookworm), siehe Tabelle 1.1.

Debian-Basis	12 (bookworm)
Linux Kernel	6.1
GNU bash	5.2.15(1)
Python	3.11.2
VLC media player	3.0.20
texlive (Latex)	2022 / Debian

Tabelle 1.1: Raspberry Pi OS Lite, Version vom 11. Dezember 2023

Python 2 gibt es nicht mehr, nur noch Python 3. Einige, früher beliebte Programme wurden ersetzt. Dazu gehört der Mediaplayer `omxplayer` (Ersatz: mpv oder VLC, beide mit hardware acceleration) und Programme zur Kamerasteuerung (raspistill and raspivid, ersetzt unter anderem durch libcamera-jpeg und libcamera-vid).

Der neue, für die Ausgabe von Video und Audio über HDMI zuständige, display driver `vc4-kms-v3d` wird im Raspberry Pi OS ab der Version vom 30. Oktober 2021 eingesetzt. Leider: „this has come with its own set of challenges“. Inzwischen läuft die Videoausgabe mit diesem Treiber meistens („it works on my machine“), aber er hat auch einen eigenen HDMI Audiotreiber, welcher (zumindest in der Lite Version des Betriebssystems) einige Schwierigkeiten bereiten kann. Die einfachste Lösungsmöglichkeit, die auch diesem Manuskript zugrund liegt, ist der Ersatz des display drivers `vc4-kms-v3d` durch `vc4-fkms-v3d` (mit einem `f` wie fake). Dies wird im Abschnitt „Anpassung des display drivers in der Datei `/boot/config.txt`“ beschrieben (siehe Seite 46).

Alternativ kann man eine alte Version des Betriebssystems verwenden, die als Raspberry Pi OS (Legacy) angeboten wird, aber das empfehle ich nicht. Andererseits kann es helfen, muss aber nicht, wenn man `vc4-kms-v3d` zusammen mit dem sound server Pipewire (siehe Seite 78) oder dem (älteren) sound server PulseAudio benutzt. Ein entsprechendes Paket für den sound server muss dann installiert werden. Der Begriff `sound server` wird später erläutert (siehe Abschnitt 2.2.1 auf Seite 67).

1.1.3 Wichtige Begriffe

Datei

Eine Datenmenge auf einem Speichermedium, die als logische Einheit betrachtet wird, heißt Datei (englisch: file) und erhält einen eindeutigen Namen (filename), bei dem Groß- und Kleinschreibung unterschieden werden. In Linux ist jede Datei einem Nutzer zugeordnet, dem owner (Eigentümer).

Der Name einer Datei kann in Linux aus bis zu 255 Zeichen bestehen, wobei Sonderzeichen und Leerzeichen vermieden werden sollen, da es damit Schwierigkeiten geben

kann. Es gibt (im Gegensatz zu Microsoft Windows) kein System für bestimmte Datei-Endungen und der Punkt (.) kann beliebig oft im Namen vorkommen. Dateien, deren Name mit einem Punkt beginnt, sind versteckte Dateien, die beim einfachen Auflisten der Dateien eines Verzeichnisses nicht angezeigt werden (mit dem Befehl `ls` als Eingabe in einer Shell, siehe Seite 113). Mehr über Dateinamen findet man auf Seite 92.

Dateien sind, gemäß einer internationalen Vereinbarung (Filesystem Hierarchy Standard = FHS) in einer Baumstruktur von Verzeichnissen (directories) geordnet.[4] Das erste Verzeichnis (directory), die Wurzel des Verzeichnisbaums, nennt man root (`/`).[5]

In Linux-Systemen findet man als Unterverzeichnisse unter anderem `boot` (mit Dateien, die für den Start des Betriebssystems gebraucht werden), `bin` (mit Dateien des Betriebssystemkerns, die ausführbare Programme, also mögliche Prozesse, enthalten), `etc` (mit Konfigurationsdateien; das sind Dateien, die Einstellungen und Werte für wichtige Programme speichern), `root` (mit den Dateien, die dem Systemadministrator zugeordnet sind), und `usr` (mit Dateien, die wichtige ausführbare Programme enthalten, die aber nicht dem Betriebssystemkern zugeordnet sind).

Programm und Prozess

Ein *Programm* ist eine Sammlung von Befehlen an den Rechner, die in einer oder mehreren Dateien gespeichert sind. Ein Programm das in den Arbeitsspeicher geladen wurde und vom Rechner ausgeführt wird, heißt *Prozess*. In einem Mehrbenutzersystem (multiuser system) wie Linux werden mehrere Prozesse gleichzeitig ausgeführt.

Die Prozesssteuerung durch das Betriebssystem startet und beendet Prozesse und verteilt den Arbeitsspeicherplatz sowie die Rechenzeit des Prozessors. Am Anfang, wenn das Betriebssystem geladen wird, startet es den init-Prozess. Ein Prozess kann neue Prozesse starten. Nach dem Start des Betriebssystems wird jedem neuen Prozess eine eindeutige Identifikationsnummer, die Process ID (PID = process identifier), zugeteilt. Einen Überblick über die laufenden Prozesse erhält man mit dem Befehl `pstree`.

Jeder Prozess ist einem Nutzer zugeordnet. Prozesse des Kernels (Kern des Betriebssystems) sind `root` zugeordnet. Für die Ein- und Ausgabe vom und an den Nutzer gibt es Datenströme (siehe unten), die Standardeingabe (0, Tastatur), die Standardausgabe (1, meistens der Bildschirm) und die Standardfehlerausgabe (2, Bildschirm).

Prozesse können, müssen aber nicht, von der Shell (bei uns in der Regel die `bash`, siehe Seite 107) verwaltet werden. Dies geschieht zusätzlich zur Prozessverwaltung durch den Kernel des Betriebssystems. Von der Shell verwaltete Prozesse heißen *jobs*.

[4] Es ist üblich, aber irreführend, den Dateibaum als hierarchische Struktur zu bezeichnen. In einer Hierarchie gehört jedes Element zu einer Rangstufe, die ein wesentliches Merkmal jedes Elements ist. Namensgebend ist das System der Weihe-Stufen in der katholischen Kirche (Bischöfe, Priester, Diakone, Laien). Altgriechisch hieros = heilig und arche = Amt, Herrschaft.

[5] Leider ist „root“ mehrdeutig. Es bezeichnet a) die Wurzel des Verzeichnisbaums, b) den Systemadministrator und c) das Verzeichnis, das dem Systemadministrator zugeordnet ist.

home directory (Heimatverzeichnis)

Linux ist ein Mehrbenutzersystem, in dem jedem Nutzer ein besonderes Verzeichnis dauerhaft zugeordnet wird, nämlich das home directory (deutsch: Heimatverzeichnis oder Benutzerverzeichnis). Dateien, die einem Nutzer gehören, zum Beispiel mit eigenen Programmen, aber auch Konfigurationsdateien des Nutzers, werden normalerweise in seinem home directory gespeichert und der Nutzer entscheidet, ob er anderen Nutzern die Verwendung einer dieser Dateien erlauben will. Unmittelbar nach der Anmeldung eines Nutzers (login) wird sein home directory zum Arbeitsverzeichnis.

In den meisten Linux-Systemen sind die home directories der normalen Nutzer Unterverzeichnisse des Verzeichnisses `/home/`. Das home directory des Nutzers `pi` ist dann `/home/ahg/` (nicht das Verzeichnis `/home/`). In manchen Linux-Systemen wird für das Verzeichnis `/home/` die Einrichtung einer eigenen Partition empfohlen, aber nicht für Einzelplatz-Rechner mit Debian, und auch nicht in unserem Raspberry Pi OS Lite. Das Nutzerkonto der Betriebssystemverwaltung, `root`, hat ein besonderes home directory, nämlich `/root/`. Es ist ein Unterverzeichnis des root directory `/`.

Der Pfad zum home directory ist für den aktuellen Nutzer in der Umgebungsvariablen `$HOME` gespeichert. Wenn der aktuelle Nutzer `pi` in der Kommandozeile den Befehl

```
echo $HOME
```

eingibt, erhält er `/home/ahg` als Antwort. In der Kommandozeile steht auch das Zeichen ∼ (Tilde) für das home directory des aktuellen Nutzers und ∼`hugo` steht für das home directory des aktuellen Nutzers `hugo`. In der Kommandozeile macht der einfache Befehl

```
cd
```

das home directory des aktuellen Nutzers zum Arbeitsverzeichnis.

Datenstrom

Ein *Datensatz* ist eine geordnete Menge (im mathematischen Sinn) von Daten. In digitalen Computern ist ein *Datenstrom* (englisch data stream, kurz stream) die stückweise Übertragung eines großen Datensatzes Q, wobei sich das übertragene Stück $S \subseteq Q$ mit der Rechenzeit ändert, welche einem diskreten Takt folgt.

Eingaben an ein Programm, zum Beispiel über die Tastatur, stellen einen eingehenden Datenstrom (downstream) dar. Ausgaben über den Bildschirm sind ein ausgehender Datenstrom (upstream). Nicht ganz passend werden Datenströme mit dem Fluss von Wasser veranschaulicht. Ein Datenstrom hat eine Quelle (englisch: source) und eine Senke (sink). Wenn eine Datei gelesen wird, ist sie Quelle eines Datenstroms. Wird sie geschrieben, ist sie eine Senke.

Datenströme sind nicht kontinuierlich in der Zeit und es gibt kein beliebig teilbares Datenkontinuum. Die kleinste Einheit binärer Daten ist ein Bit. Meistens werden, um effizient zu arbeiten, größere Datensätze in einem Takt übertragen. In manchen Datenströmen kann man vor und zurück springen, zum Beispiel beim Lesen einer Datei.

Bei Datenströmen, deren Quelle nicht vom Programm gesteuert wird, zum Beispiel bei Eingaben über die Tastatur, geht das nicht.

In Linux (oder allgemeiner: in Unix-artigen Betriebssystemen) gibt es drei Standard-Datenströme (standard streams), nämlich `stdin` (Eingabe von der Tastatur in ein Programm), `stdout` (reguläre Ausgabe von einem Programm auf dem Bildschirm) und `stderr` (Fehlerausgabe von einem Programm auf dem Bildschirm). Jeder Dateistrom braucht einen geöffneten Kanal. Die Kanäle der Standard-Datenströme sind automatisch geöffnet. Andere Kanäle, zum Beispiel zum Lesen oder zum Schreiben einer Datei, müssen ausdrücklich geöffnet werden, bevor Daten fließen können, und sollten danach geschlossen werden.

Quellen oder Senken von Datenströmen können Dateien sein oder ähnlich wie Dateien behandelt werden. Einer Datei, für die ein Datenstrom geöffnet ist, wird vom Betriebssystem eine Zahl eindeutig zugeordnet, die *file descriptor* genannt wird. Dateien und dateiähnliche Objekte können als Quelle, Senke oder gleichzeitig als Quelle und Senke von Datenströmen dienen.

Für Prozesse werden drei besondere Datenströme für die Interaktion mit dem Nutzer zur Verfügung gestellt, deren Daten Zeichen darstellen, welche vom Nutzer gelesen werden können (zeichenorientiert): a) die Standardeingabe, deren Quelle den file descriptor 0 hat und die normalerweise mit der Tastatur verbunden ist, b) die Standardausgabe, deren Senke den file descriptor 1 hat und die normalerweise mit dem Bildschirm verbunden ist, und c) die Standardfehlerausgabe deren Senke den file descriptor 2 hat und die normalerweise mit dem Bildschirm verbunden ist. Die Trennung der Datenströme für gewöhnliche Aufgaben und Fehlermeldungen hat sich als sinnvoll erwiesen.

Als Quelle der Standardeingabe kann statt der Tastatur eine beliebige zeichenorientierte Datei dienen. Ebenso kann als Senke der Standardausgabe oder der Standardfehlerausgabe statt des Bildschirms eine zeichenorientierte Datei dienen. Das wird durch Umleitungen erreicht, über die wir später sprechen werden.

Daemon

Ein Daemon ist ein Programm, das im Hintergrund ohne direkte Kommunikation mit dem Nutzer abläuft und wartet, bis ein anderes Programm einen Auftrag erteilt. Ein Beispiel ist der Druckerdaemon (printer daemon) lpd, der (zum Beispiel von einem Textverarbeitungsprogramm) Druckaufträge entgegennimmt (auch wenn der Drucker noch ausgeschaltet ist) und speichert, bis er sie drucken kann. Daemonen (die etwa den Diensten oder services bei Microsoft-Windows entsprechen) werden meistens vom Betriebssystem beim Systemstart geladen und verbrauchen auch beim Warten etwas von der Rechnerleistung.

Terminal

Ein Computerterminal, kurz Terminal, war ursprünglich ein eigenständiges Gerät zur Ein- und Ausgabe von Daten in einem Computersystem. Der Name stammt vom lateinischen *terminus*, Grenzstein, da Terminals die Schnittstellen zu den Benutzern des

Computersystems bildeten. Daher kommt auch die deutsche Bezeichnung als Datenendgerät. In modernen Computern benutzt man stattdessen in der Regel eine Tastatur (Gerät zur Dateneingabe) und einen Bildschirm (Gerät zur Datenausgabe).

Die Funktion eines Terminals wird heute mit einem Programm namens Terminalemulator bereitgestellt. Meistens dient der Terminalemulator nur zur Ein- und Ausgabe von Text. Es ist üblich, und wir werden das auch so machen, den Textterminalemulator kurz Terminal zu nennen. Auch ein Terminalprozess (laufendes Terminalprogramm) wird kurz als Terminal bezeichnet. Die Bedeutung von *Terminal* hängt also vom Kontext ab.

In Linux-Systemen gibt es verschiedene Terminals. Ohne und mit graphischer Oberfläche stellt der Linux-Kernel mehrere sogenannte *virtuelle Konsolen* zur Verfügung, mit Namen wie `tty1`, `tty2`, und so weiter. Sie werden virtuell genannt, weil man mit nur einer Tastatur und nur einem Bildschirm alle virtuellen Konsolen verwenden kann. Es sind also nicht mehrere Konsolen physisch vorhanden. Die Bezeichnung „tty“ ist eine Abkürzung von englisch teletypewriter oder teletype, auf deutsch Fernschreiber. Fernschreibgeräte dienten als Terminal, bevor es Bildschirme gab. Seit Dezember 2006 findet man sie nicht mal mehr bei der Bundeswehr, allerdings noch in anderen Museen. Uralte Geräte, zum Beispiel Fernschreiber, können noch immer mit Linux betrieben werden und arbeiten dann als physische Konsole im Text-Modus, ohne Graphik, durch Austausch von Textzeichen mit dem Betriebssystem.

In Linux-Betriebssystemen mit graphischer Oberfläche (Desktop) läuft im Hintergrund das X Window System (X). Es wird vom Betriebssystem in einer virtuellen Konsole gestartet, was aber der Benutzer in der Regel nicht bemerkt. Man kann aber auch in eine andere virtuelle Konsole wechseln, die ohne X arbeitet. Wie das geht, hängt von der Betriebssystemvariante ab. Mehr zu den virtuellen Konsolen im Betriebsstem Raspberry Pi OS findet man im Abschnitt 3.2.1 auf Seite 83.

Für Linux-Betriebssystemen mit graphischer Oberfläche gibt es, zusätzlich zu virtuellen Konsolen, weitere Terminalprogramme, zum Beispiel GNOME Terminal, LXTerminal und XTerm. Deren Grundfunktionen sind gleich, aber sie unterscheiden sich in Zusatzfunktionen und Erscheinungsbild. Verzichtet man auf besondere Zusatzfunktionen, ist XTerm eine gute Wahl, weil er auf allen Linux-Systemen mit graphischer Oberfläche verfügbar ist und man so stets ein vertrautes Erscheinungsbild erhält und das Verhalten des Terminals kennt. Wer ein Linux-System mit X benutzt, kann die in diesem Manuskript beschriebenen Programme ebenfalls in einem Terminalprogramm wie XTerm benutzen.

Booten

Durch Anlegen der Spannungsversorgung (power on) wird der Raspberry Pi eingeschaltet und es folgt das *Booten*, das heißt das Hochfahren oder Starten des Betriebssystems, verschieden von anderen Rechnern. Zunächst ist nur die GPU (graphics processing unit) und danach die CPU (central processing unit) beteiligt (außer beim Modell Pico).

Die GPU liest automatisch einen in Maschinensprache geschriebenen Programmcode vom eingebauten ROM und dann folgen mehrere Stufen, bis der Linux kernel (kernel.img) in den Arbeitsspeicher geladen ist.

Ab hier übernimmt die CPU das Booten und startet `systemd` als erstes Programm.

Ein laufendes Programm heißt Prozess und hat eine PID (process identifier). `systemd` hat die PID 1. Andere Programme werden direkt oder indirekt von `systemd` gestartet und beendet, auch beim *shutdown* (Herunterfahren des Betriebssystems).

systemd

Das Programmpaket `systemd` ist Teil der Betriebssysteme Debian und Raspberry Pi OS, sowie anderer Linux-Distributionen. Wesentlicher Teil ist der gleichnamige Dämon, der beim Booten als erster Prozess (mit der PID 1) andere Prozesse, insbesondere auch andere Dämonen, startet und damit das früher verwendete System-V-Init-System ersetzt. Um auf einem Linux-Rechner zu prüfen, ob `systemd` läuft, gibt man den Befehl

```
ps --no-headers -o comm 1
```

ein und erhält die Antwort `systemd`, wenn dieser Dämon umherläuft.

Die von `systemd` verwalteten Betriebsmittel (resources) des Rechners werden (units) genannt. Dazu gehören unter anderem Dienste (services), Geräte (devices), Einhängepunkte des Dateisystems (mount points) und Zeitsteuerungen (timers).

Zu jeder unit gibt es (mindestens) eine Konfigurationsdatei, die als *unit file* bezeichnet wird und eine Dateinamenendung hat die auf den Typ der unit hinweist, zum Beispiel die Endung `.service` für einen Dienst oder `.timer` für eine Zeitsteuerung. `systemd` steuert auch das Anmelden eines Nutzers (user), das *Login*, welches vom Programm `agetty` durchgeführt wird. Für die automatische Anmeldung (Autologin) gibt es einen Dienst mit dem unit file `autologin@.service`.

Die unit files liegen entweder im Verzeichnis `/lib/systemd/system` (für units des Betriebssystems) oder im Verzeichnis `/etc/systemd/system` (für units der Nutzer). Die unit files der Nutzer können mit Administratorrechten in einem Texteditor bearbeitet werden. In der Kommandozeile kann man auf `systemd` mit dem Befehl `systemctl` zugreifen, der zahlreiche Optionen hat und Teilbefehle für verschiedene units bereitstellt.

Ctrl-C und Ctrl-D

Auf jeder Computertastatur (englisch keyboard) gibt es unter anderem die Taste `Ctrl`, die bei deutschen Tastaturen auch mit `Strg` beschriftet sein kann. Das gleichzeitige Drücken der `Ctrl`-Taste und der mit `C` beschrifteten Taste (die ohne weiteren Tastendruck den Kleinbuchstaben `c` ergibt), nennen wir `Ctrl-C` oder `Ctrl-c`. Diese Tastenkombination sendet ein Signal zu Unterbrechung (interrupt) und erlaubt es oft, ein laufendes Programm (Prozess) abzubrechen und zur Kommandozeile zurückzukehren.

In einem Linux Betriebssystem ohne graphische Oberfläche (X) gibt es keine allgemeine „Zwischenablage“ (englisch clipboard) und in der Regel kein „Copy and Paste“ mit den Tastenkombinationen `Ctrl-C` und `Ctrl-V`. Stattdessen kann man in der Kommandozeile einen Datenstrom in eine beliebige Datei lenken oder umlenken, und diese Datei später an eine andere Datei anhängen und so Daten übertragen. Außerdem kann man eine Maus für allgemeines „Copy and Paste“ benutzen (siehe Seite 48). Das Programm

`wordgrinder` ist eine Ausnahme: Es erlaubt „Copy and Paste“ mit `Ctrl-C` und `Ctrl-V` (siehe Seite 236).

Das gleichzeitige Drücken der `Ctrl`-Taste und der mit `D` beschrifteten Taste (die ohne weiteren Tastendruck den Kleinbuchstaben `d` ergibt), nennen wir `Ctrl-D` oder `Ctrl-d`. Diese Tastenkombination wird von vielen Prozessen als EOF Signal erkannt und als Wunsch zum Verlassen verstanden. Insbesondere dient es zum Verlassen des interaktiven Modus von Python (der mit `python -i` gestartet wird) und anderen Programmen (zum Beispiel `bc`, siehe Seite 224).

1.2 Hardware

1.2.1 Raspberry Pi

Der Raspberrry Pi ist ein *single board computer* (Kleinrechner auf einer Platine). Es gibt verschiedene Modelle; Tabelle 1.2 fasst Eigenschaften einiger neuer Modelle zusammen. Abbildung 1.2 zeigt ein Schema des Typs *Pi 3 Model B*. Alle Prozessoren haben kein Hyper-Threading, das heißt: die Zahl der logischen oder virtuellen Kerne ist gleich der Zahl der physischen Kerne. Wenn beim Start des Betriebssystems oben auf dem Bildschirm Himbeeren (raspberries) gezeigt werden, entspricht ihre Anzahl der Zahl der Prozessorkerne des Raspberry Pi.

Modell	Jahr	SoC	CPU / Kerne	SDRAM	USB 2	USB 3
RPi 3 Model B	2016	BCM2837	Cortex-A53 / 4	1 GiB	4	0
RPi 3 Model B+	2018	BCM2837B0	Cortex-A53 / 4	1 GiB	4	0
RPi 4 Model B	2019	BCM2711	Cortex-A72 / 4	1 – 4 GiB	2	2
	2020			8 GiB		

Tabelle 1.2: Eigenschaften neuerer Modelle des Raspberry Pi, Modellreihe B.

In den meisten Fällen gilt: Eine microSD-Karte (oder in den ältesten Modellen eine SD-Karte) dient als Datenspeicher für Betriebssystem, Programme und Daten. Ein USB-Kabel versorgt den Raspberrry Pi mit elektrischer Energie und ein HDMI-Kabel verbindet ihn mit einem Monitor. Über USB sind Tastatur (und Maus) angeschlossen. LAN, und bei neueren Modellen auch WLAN und Bluetooth sind eingebaut. Zum Anschluss externer Geräte gibt es USB-Buchsen vom Typ 2.0 oder 3.0 (blau gekennzeichnet).

Der Raspberrry Pi wird durch Bereitstellung der Energieversorgung eingeschaltet. Eine rote LED (namens PWR) zeigt dies an und das Betriebssystem startet (nachdem es auf der Speicherkarte installiert wurde). Zum Ausschalten wird das Betriebssystem heruntergefahren und *danach* trennt man den Raspberrry Pi von der Energieversorgung.

Vier beziehungsweise fünf kleine Leuchtdioden (LED) auf der Platine informieren über den Zustand (Status) des Raspberry Pi (siehe Tabelle 1.3).

Analoge Ausgänge für Audio und Video sind bei neueren Modellen des Raspberry Pi in einer TRRS-Buchse zusammengefasst. Hier kann man einen vierpoligen Klinkenstecker

Abbildung 1.1: Single Board Computer (SBC) Raspberry Pi 2 B. Der Urheber, Evan-Amos, hat das Bild gemeinfrei gegeben (public domain); Wikimedia Commons, Datei Raspberry-Pi-2-Bare-BR.jpg . Oben links befindet sich die Stiftleiste J8. Die äußere, im Bild obere Reihe der Pins von J8 hat geradzahlige Nummern (im Bild links mit `2` beginnend und rechts mit `40` endend). Die innere, im Bild untere Reihe hat ungeradzahlige Nummern (im Bild links mit `1` beginnend und rechts mit `39` endend.)

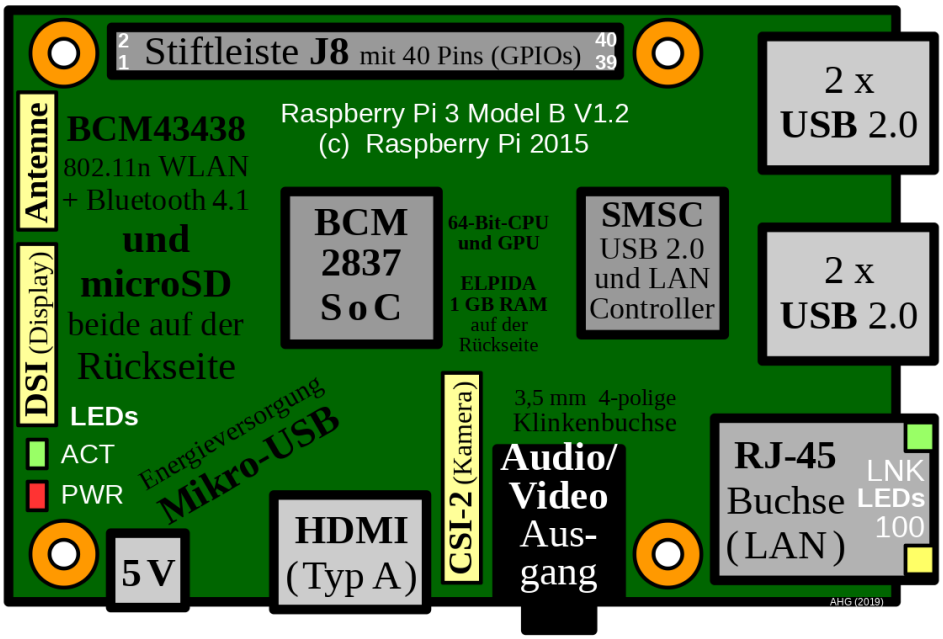


Abbildung 1.2: Schema des Single Board Computers (SBC) Raspberry Pi 3 B

Name	Farbe	Funktion der LED
ACT	grün	blinkt beim Zugriff auf die SD-Karte, von Programmen schaltbar
PWR	rot	leuchtet, wenn Versorgungsspannung (auf der 3,3 V Leitung) anliegt
FDX	grün	leuchtet, wenn eine Full Duplex-Verbindung zum LAN besteht
LNK	grün	leuchtet bei Netzwerkverbindung, blinkt bei LAN-Datenverkehr
100	gelb	leuchtet bei LAN-Übertragungsrate 100 MBit/s (Fast Ethernet)

Tabelle 1.3: Kontroll-LED im Raspberry Pi 1 A und 1 B. Die LED `FDX` fehlt bei den Modellen 1 B+, 2 B, 3 B, 3 B+ und 4 B. Beim Raspberry Pi 3 B, Pi 3 B+ und Pi 4 B befinden sich die LED `ACT` und `PWR` in einer Ecke der Platine und die LED `LNK` und `100` innerhalb der RJ-45-Buchse.

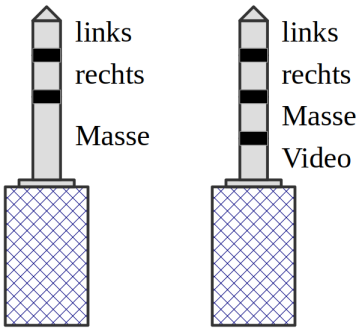


Abbildung 1.3: Anschlüsse eines Klinkensteckers für den 3,5 mm Analogausgang. Links: Stecker für den Audio-Ausgang des Raspberry Pi Modells 1B , rechts: Stecker für den Audio/Video-Ausgang der Modelle 2B, 3B und 4B.

(Durchmesser 3,5 mm) anschließen. Man muss aber auf die Signalbelegung achten, siehe Abbildung 1.3 und Tabelle 1.4. Andere Geräte haben möglicherweise eine andere Belegung. Wo beim Raspberry Pi der Videoausgang ist, ist bei einem modernen Smartphone (mit TRRS-Buchse gemäß CTIA) oft ein Mikrophoneingang.[6] Beim alten Raspberry Pi 1B gibt es eine (gelbe) Cinch-Buchse für das Bild- und eine dreipolige Klinkenbuchse (Stereo, 3,5 mm Durchmesser) für das Tonsignal.

Spitze	Ring 1	Ring 2	Schaft
Audio links	Audio rechts	Signalmasse	Video

Tabelle 1.4: Belegung der Audio/Videosignale in der TRRS-Buchse ab Modell 2

[6] A raspberry is not an apple.

Energieversorgung

Die Leistungsaufnahme hängt von der Belastung durch die Elektronik des Raspberry Pi selbst (SBC) und von den angeschlossenen Peripheriegeräten ab. Ab Modell 2 B steigt die elektrische Leistungsaufnahme im SBC neuerer Modelle der B-Modellserie. Tabelle 1.5 listet die Leistung auf, die für verschiedene Modelle des Raspberry Pi bereitgestellt werden sollte, wenn die angeschlossenen Peripheriegeräte nicht zu leistungshungrig sind.

Der tatsächliche Bedarf kann höher sein und erfordert dann ein stärkeres Netzteil. Allerdings ist die Leistungsabgabe über den USB-Bus begrenzt. Das erste B-Modell wird hier 1 B genannt, obwohl es eigentlich Raspberry Pi Model B (ohne Nummer) heißt.

	1 B	2 B	3 B	4 B
Leistung	5 W	5 W	7,5 W	10 W
Strom	1 A	1 A	1,5 A	2 A

Tabelle 1.5: Elektrische Leistung, die für den Raspberry Pi bereitgestellt werden sollte

Die Energieversorgung erfolgt über eine USB-C Buchse (Modell 4 B) oder eine Mikro-USB-B Buchse (frühere Modelle) im Raspberry Pi. Die Buchse ist bei USB-C symmetrisch bezüglch einer Drehung um 180°, bei Mikro-USB jedoch nicht. Das Netzgerät muss einen entsprechenden Stecker haben und eine Ausgangsspannung von mindestens 5 V und höchstens 5,25 V liefern. Zu beachten ist, dass viele Netzgeräte die vom Hersteller angegebenen Sollwerte in der Praxis nicht erreichen. Im Zweifelsfall muss man das Netzteil selbst prüfen und die Spannung unter Belastung messen.Ein Netzteil, das (bei richtiger Ausgangsspannung) mehr Strom liefern kann als notwendig, ist unschädlich.

Bei den Modellen 3 B und 3 B+ wird der über die Mikro-USB Buchse zugeführte Dauerstrom (holding current) auf höchstens 2,5 A begrenzt. Dies wird durch eine selbst zurückstellende Sicherung, kurz Polyfuse genannt, gewährleistet (Typ: MF-MSMF250/X oder MF-MSMF250/16X für eine Maximalspannung von 16 V). Bei höheren Strömen erwärmt sich die Sicherung und unterbricht irgendwann den Stromkreis. Bei 5 A (tripping current) geschieht dies innerhalb von 5 s. Die Polyfuse befindet sich auf der Rückseite der Platine in der Nähe der Mikro-USB Buchse. Die Modelle 1 B und 2 B sind für eine geringere Leistungsaufnahme ausgelegt und enthalten daher eine 0,75 A Polyfuse beziehungsweise eine 2,0 A Polyfuse (holding current). Bei Modell 4 B gibt es die Polyfuse nicht mehr.

Man liest oft, dass die Netzteile eine Ausgangsspannung von 5 V haben sollen. In der Praxis ist es aber besser, wenn sie 5,1 V oder 5,2 V beträgt. Intern braucht das SoC zwar 5 V, aber ein bisschen Schwund ist immer. Wenn die Polyfuse einen Innenwiderstand von 0,05 Ω hat, fällt bei einem Strom von 2 A eine Spannung von 0,1 V ab.

Mit Netzgeräten nicht ausreichender Leistung arbeitet der Raspberry Pi fehlerhaft, insbesondere beim Anschluss weiterer Geräte über die USB-Buchsen. Für einen mobilen Einsatz kann ein geladener Akku angesteckt werden. Im Auto kann der Raspberry Pi über eine Bordspannungssteckdose und einen Adapter, der 5 V – 5,2 V Spannung erzeugt, betrieben werden.

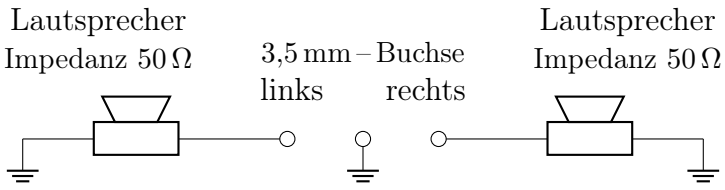


Abbildung 1.4: Hochohmige Lautsprecher zur Lastanpassung an den Audio-Ausgang

Die neueren Modelle des Raspberry Pi (ab 2 B) haben einen Spannungssensor, der ab einer Spannung von etwa 4,6 V ein Signal gibt. Dieses steuert bei den neueren Modellen die rote PWR-LED (siehe Tabelle 1.3). Sie leuchtet also nur bei ausreichender Spannung.

Ein aktiver USB-Hub, der weitere USB-Buchsen mit eigener Stromversorgung bereitstellt, ist eine sinnvolle Ergänzung. Damit können auch externe Geräte, die viel elektrische Leistung benötigen, zum Beispiel externe Festplatten oder DVD-Laufwerke, angeschlossen werden.

Aus der Spannungsversorgung von etwa 5 V werden mit Spannungsreglern auf der Platine 1,8 V und 2,5 V und 3,3 V für bestimmte Bauteile erzeugt.

Audio-Ausgänge

Alle Modelle des Raspberry Pi haben mindestens einen HDMI-Ausgang und stellen damit einen digitalen Audioausgang bereit. Die meisten Modelle haben auch einen analogen, 2-kanaligen (Stereo) Audio-Ausgang über eine dreipolige TRS oder eine vierpolige TRRS Buchse. Letztere überträgt zusätzlich ein Video-Signal.

Wenn man nur das Audio-Signal benötigt, kann man an die vierpolige TRRS-Buchse einen dreipoligen Klinkenstecker (TRS) anschließen. Dabei wird der Videoausgang vom Stecker mit Masse verbunden (kurzgeschlossen), aber das ist diesem Fall bedacht und unschädlich.

Der analoge Audioausgang ist zu hochohmig, um einen niederohmigen Lautsprecher (4 Ω – 8 Ω) ohne Verstärker anzuschließen. Es wurde berichtet (Burngate, https://www.raspberrypi.org/forums/viewtopic.php?t=225118), und ich gebe dies ungeprüft ohne Gewähr weiter, dass beim Raspberry Pi 3B und 3B+ die Ausgangsimpedanz eines Kanals 68 Ω beträgt und die Spannung maximal 1 V p-p (Spitze-Tal-Wert). Der Audioausgang hat also eine viel höhere Impedanz als ein üblicher Audio-Verstärker.

Wenn man an jedem Audiokanal (links und rechts) einen 50 Ω oder 100 Ω Lautsprecher anschließt (siehe Abbildung 1.4), kann man in einem ruhigen Raum auch ohne Verstärker schon gut hören (stereo). 50 Ω Einbau-Lautsprecher und dreipolige 3,5 mm Einbaubuchsen sind im Elektronik-Versandhandel erhältlich.

USB-Soundkarten bieten, abgesehen von der Signalverzögerung (time lag) der USB-Geräte, eine bessere Audio-Qualität als der oben beschriebene Analogausgang. Außerdem haben sie meistens auch einen Mikrophonanschluss. Es gibt auch Wandler, die das digitale Audiosignal des HDMI-Ausgangs in ein analoges Audiosignal wandeln.

Der digitale Audioausgang über HDMI und der analoge Ausgang über die TRS oder TRRS Buchse können nicht als Eingänge dienen. Wie oben gesagt, können jedoch USB-

Soundkarten als Audioeingang dienen. Die gleichzeitige Nutzung von digitalen und analogen Audioausgängen ist möglich (anders als bei den Videoausgängen).

Video-Ausgänge

Am Raspberry Pi wird ein Bildschirm (Monitor) meistens über das digitale Video-Signal eines HDMI Ausgangs angeschlossen. Einige Modelle haben auch einen analogen Video-Ausgang, entweder als Cinch-Buchse oder als ein Kanal in einer TRRS-Buchse.

Zero Modelle haben einen Mini-HDMI (Typ C) Ausgang, die Modelle 1 bis 3 haben einen HDMI (Typ A) Ausgang, und das Modell 4 B hat zwei Micro-HDMI (Typ D) Ausgänge. Wenn der Monitor nur einen DVI-Eingang hat, kann man ein HDMI/DVI-Adapterkabel benutzen. Allerdings wird damit nur das Video-Signal übertragen, nicht das Audio-Signal.

Das analoge Video-Signal heißt Composite Video (CVBS) oder Farb-Bild-Austast-Synchron-Signal (FBAS). Es hat eine schlechtere Auflösung und wird heute nur noch selten benutzt. Wenn man es verwenden will, muss man die richtige Variante einstellen (Parameter `sdtv_mode` 0 = NTSC, 1 = NTSC-J, 2 = PAL, 3 = PAL-N / PAL-CN). In Deutschland wird PAL verwendet. Außerdem kann man das Bildformat wählen (Parameter `sdtv_aspect`) und bei Bedarf die Farbsignalverarbeitung ändern (Parameter `sdtv_disable_colourburst`) oder das Bild drehen (Parameter `display_rotate`). Bei Modellen mit TRRS-Buchse kann man ein Adapterkabel anschließen, das in drei Cinch-Steckern endet (für Composite Video = FBAS, linken und rechten Audiokanal; gemeinsame Signalmasse). Man beachte dabei, dass die Belegung der vier Pole nicht einheitlich festgelegt ist. Für Mobiltelephone gibt es die Belegungsvarianten OMTP und CTIA. Die TRRS-Buchse eines Raspberry Pi braucht ein Adapterkabel vom Typ CTIA. Im Zweifelsfall muss man messen, wo welche Signale ankommen.

Die genannten Videoausgänge können nicht als Eingang dienen. Die bisher benutzte Firmware (siehe Seite 126) gestattet nicht, dass HDMI und der analoge Videoausgang gleichzeitig eingeschaltet sind. Es gibt auch displays (Bildschirme), die über GPIO (siehe Seite 19) an einen Datenbus des Raspberry Pi angeschlossen werden.

LAN oder WLAN, Kameras

Zugang zum Internet kann man über LAN oder WLAN (englisch meist Wi-Fi genannt) erreichen. Zum Anschluss über LAN genügt es, das entsprechende Kabel an die RJ-45 Buchse anzuschließen (siehe Abbildung 1.2). Der Anschluss über WLAN wird später, ab Seite 53, beschrieben.

Bei Kameras unterscheiden wir universell einsetzbare Webcams, die über USB oder WLAN angeschlossen werden, und besondere Raspberry Pi Kameras (Pi Cameras), die über ein 15-adriges Flachbandkabel (englisch ribbon cable) mit dem CSI-2 Verbinder (Typ: zero insertion force electrical connector) auf der Platine des Raspberry Pi verbunden werden (siehe Abbildung 1.2). Dafür muss der Klemmsteg auf dem CSI-2 Verbinder hochgezogen, das Kabel eingeschoben und der Steg danach niedergedrückt werden. Der mechanische Anschluss einer Pi Camera an den CSI-2 Verbinder wird in einem

YouTube-Video von Liz Upton (Raspberry Pi Foundation) gezeigt. Das Objektiv einer neuen Kamera wird von einer Schutzfolie bedeckt, die vor dem Betrieb entfernt wird.

Die Steuerung einer Pi Camera oder einer USB-Webcam mit Programmen wird ab Seite 175 beziehungsweise ab Seite 177 erörtert.

1.2.2 Datenaustausch durch GPIO und Datenbusse

Zum Datenaustausch mit anderen Geräten haben neuere Modellen des Raspberry Pi 40 elektrische Anschlusspunkte, die auf Pins der Stiftleiste (englisch: pin header oder header) J8 liegen. Die ersten Modelle, 1 A und 1 B, hatten nur 26 Pins auf der entsprechenden Stiftleiste P1.

Die GPIO des Raspberry Pi können auch Datenströme senden oder empfangen. Für einfache Datenübertragungsaufgaben gibt es verschiedene serielle Bussysteme, die zum Teil von Raspberry Pi unterstützt werden. Dazu gehören der Eindraht-Bus (1-Wire), der Datenbus SPI und der der Datenbus I^2C.

Anschlusspunkte mit konstanter Spannung

Alle Spannungen beziehen sich auf die Signalmasse (GND). Einige Anschlusspunkte stellen die Signalmasse bereit, siehe Tabelle 1.6.

GND							
Pin 6	Pin 9	Pin 14	Pin 20	Pin 25	Pin 30	Pin 34	Pin 39

Tabelle 1.6: Pins der Stiftleiste J8, die auf Signalmasse (GND) liegen

Andere Anschlusspunkte bieten eine feste positive Spannung, siehe Tabelle 1.7.

3,3 V	
Pin 1	Pin 17

5 V	
Pin 2	Pin 4

Tabelle 1.7: Anschlusspunkte mit fester positiver Spannung von 3,3 V oder 5 V

GPIO

Weitere Anschlusspunkte können digitale Daten (logisch: 0 oder 1) ein- und ausgeben. Sie heißen *general purpose input/output* (GPIO). Für den Datenaustausch von jeweils 1 Bit werden unter anderem die in Tabelle 1.8 genannten GPIO-Anschlusspunkte verwendet.

Jeder GPIO kann entweder zur digitalen Eingabe (input) oder Ausgabe (output) dienen. Zulässige Spannungswerte sind 0 V oder 3,3 V, welche den logischen Werten 0 (low) und 1 (high) entsprechen. Genauer gesagt wird an einem als Eingang geschalteten GPIO eine Spannung zwischen 0 V und 0,8 V als logische 0 (low) gelesen, eine zwischen 2,0 V

GPIO-22	GPIO-23	GPIO-24	GPIO-25
Pin 15	Pin 16	Pin 18	Pin 22

Tabelle 1.8: Einige der GPIO-Anschlusspunkte für 1 bit Ein- / Ausgabe

und 3,3 V als logische 1 (high). Spannungswerte zwischen 0,8 V und 2,0 V sollten nicht verwendet werden, da deren Umsetzung in einen Logikpegel unbestimmt ist.

Zu einem als Eingang geschalteten GPIO kann per Software ein interner Pullup-Widerstand oder Pulldown-Widerstand zugeschaltet werden. Deren Werte liegen zwischen 50 kΩ und 65 kΩ. Eine Ausnahme bilden GPIO 2 und 3, die beide (um ihre Nutzung für einen I2C Datenbus zu vereinfachen) einen, nicht abschaltbaren, internen 1,8 kΩ Pullup-Widerstand haben. Dies kann die Nutzung der GPIO 2 und 3 beeinflussen. Ein externer Pulldown-Widerstand wird im folgenden Abschnitt 1.2.4 verwendet.

Der Strom durch die GPIO und 3,3 V Pins darf insgesamt maximal 50 mA betragen. Durch einen einzelnen GPIO sollten höchstens 16 mA fließen. Die 5 V Pins können zusammen soviel Strom an externe Geräte abgeben, wie das Netzteil liefert, abzüglich des Stroms, der vom Raspberry gebraucht wird, vorausgesetzt die Strombegrenzung durch die eingebaute Sicherung (PolyFuse) des Raspberry Pi wird nicht überschritten.

Datenbus 1-Wire

Der Eindraht-Bus (1-Wire) wurde von der Firma *Dallas Semiconductor* entwickelt, die heute zu *Maxim Integrated* gehört, einer Tochterfirma von *Analog Devices*. Er überträgt Daten seriell (nacheinander), asynchron (ohne zusätzliches Taktsignal) über nur zwei oder drei Leitungen mit einer Geschwindigkeit von normalerweise 15,4 kbps über Entfernungen bis etwa 100 m überträgt.

Es gibt dabei eine Master-Station (hier: der Raspberry Pi) und eine oder mehrere Slave-Stationen (hier: ein oder mehrere Sensoren); jede Slave-Station hat eine eindeutige, vom Hersteller vorgegebene Identifikationsnummer.[7]

Wenn der Bus mit drei Leitungen gebaut wird, gibt es neben der Masseleitung eine Datenleitung und eine für die Versorgungsspannung (zwischen 3 V und 5,5 V). Beim Aufbau mit zwei Leitungen entfällt die Versorgungsspannungsleitung und die Slave-Station (Sensor) gewinnt die notwendige Energie „parasitär“ aus der Datenleitung. Der Betrieb mit drei Leitungen ist stabiler, und wird hier daher vorgezogen, aber man braucht eine Leitung mehr. Der Name des Eindraht-Bus (1-Wire) stammt vom Aufbau mit zwei Leitungen, da dann zusätzlich zur Masse nur eine weitere Leitung gebraucht wird.

Unten wird beschrieben, wie digitale Daten eines Temperatursensors (DS18S20) über einen Eindraht-Bus von einem einzelnen GPIO empfangen werden.

[7] Für die anstößigen Bezeichnungen *master* und *slave* gibt es noch keinen allgemein angenommenen Ersatz. Python benutzt *parent* oder *main* statt *master*, und *worker* oder *helper* statt *slave*.

Datenbus SPI

Die Datenübertragung zwischen Raspberry Pi und externen Geräten, wie dem A/D-Wandler MCP3208 (siehe Seite 31), kann über einen Datenbus *Serial Peripheral Interface* (SPI) geschehen. Alle Modelle des Raspberry Pi haben mindestens einen SPI Bus (`SPI0`).

An einem Ende des SPI Busses sitzt das Kontrollgerät (Master), hier der Raspberry Pi, und am anderen Ende ein oder mehrere davon kontrollierte Geräte (Slaves), zum Beispiel der MCP3208. Für die Signalübertragung auf dem SPI Datenbus `SPI0` werden die in Tabelle 1.9 aufgeführten J8-Pins verwendet:

GPIO-8	GPIO-9	GPIO-10	GPIO-11
Pin 24	Pin 21	Pin 19	Pin 23
$\overline{\text{CS}}$	MISO	MOSI	SCLK

Tabelle 1.9: GPIO-Anschlüsse für Signalleitungen des Datenbusses SPI

Das kontrollierte Gerät (Slave) wird angesprochen, indem die Signalleitung namens *chip select*, $\overline{\text{CS}}$, von *high* (inaktiv) auf *low* (aktiv) gesetzt wird.

Die Daten werden nacheinander (seriell) mit einer Bus-Taktrate übertragen (gleichzeitig gesendet und empfangen), welche auf der Signalleitung namens *serial clock*, SCLK, vom Kontrollgerät (Master) vorgegeben wird.

Daten werden auf der Signalleitung namens *master out slave in*, MOSI, vom Master zum Slave gesendet. Umgekehrt werden Daten auf der Signalleitung namens *master in slave out*, MISO, vom Slave zum Master gesendet.

Datenbus I²C

Inter-Integrated Circuit oder kurz I²C (gesprochen als I-Quadrat-C) ist ein serieller Datenbus für kurze Entfernungen, der 1982 von der Firma Philips Semiconductors (heute NXP Semiconductors) entwickelt wurde.

Ähnlich wie beim SPI Bus (siehe oben, Seite 21) sitzt an einem Ende das Kontrollgerät (Master), hier der Raspberry Pi, und am anderen Ende ein oder mehrere davon kontrollierte Geräte (Slaves), zum Beispiel der Temperatursensor LM75 oder das zweizeilige Display JOY-iT 16X2 LCD MODUL, von der Firma Simac Electronics. Der Master kommuniziert mit einem bestimmten Slave, indem er ihn über die Adresse anspricht.

Für die Signalübertragung auf dem I²C Datenbus werden die in Tabelle 1.10 aufgeführten J8-Pins verwendet. Die Versorgungsspannung ($+V_S$ oder VCC genannt) hängt vom angeschlossenen Bauteil ab. Zum Beispiel wird der LM75 mit $+V_S = 3{,}3\,\text{V}$ betrieben, aber das JOY-iT 16X2 LCD MODUL benötigt $\text{VCC} = 5\,\text{V}$. Auch Bauteile, die mit 5 V versorgt werden, müssen für die Kommunikation mit dem Raspberry Pi über den I²C Bus Spannungen von 0 V oder 3,3 V verwenden.

Die Aktivierung des I²C Busses wird ab Seite 375 erklärt. Temperaturmessung mit dem Sensor LM75 am I²C Datenbus durch ein Pythonprogramm wird ab Seite 376

GPIO-2	GPIO-3	GND	3,3 V	5 V
Pin 3	Pin 5	Pin 6	Pin 1	Pin 4
SDA	SCL	GND	VCC oder $+V_S$	VCC oder $+V_S$

Tabelle 1.10: GPIO-Anschlüsse für Signale und Versorgungsspannungen beim I^2C Bus

beschrieben. Die Ansteuerung eines Displays JOY-iT 16X2 LCD MODUL über den I^2C Bus mit einem Pythonprogramm findet man ab Seite 379.

1.2.3 Elektronische Bauteile

Der Raspberry Pi ermöglicht digitale Ein- und Ausgaben (digital I/O) durch GPIO, einschließlich mehrerer Datenbusse. Diese erweitern die Funktion des Rechners mithilfe externer elektronischer Schaltungen. In diesem Abschnitt werden zunächst verschiedene elektronische Bauteile vorgestellt. Danach werden elektronische Schaltungen beschrieben. Die zugehörigen Computerprogramme werden später, in den Abschnitten der Programmiersprachen, Shell (Bash) oder Python, behandelt.

Schalter und Taster

Schalter sind elektronische Bauteile, die eine näherungsweise ideale leitende Verbindung zwischen zwei Punkten (Anschlüssen) schließen oder öffnen. Wir betrachten hier nur manuell (mit der Hand) betätigte Schalter. Ein Schalter im engeren Sinne hat mindestens zwei stabile Zustände (leitend und nicht leitend). Wenn nur ein Zustand stabil ist und der andere Zustand nur durch händische Betätigung eintritt, spricht man von einem Taster oder einer Taste. Wichtige Schaltertypen sind:

- Einschalter, Ein- und Ausschalter, (2 Anschlüsse, 2 stabile Zustände)
- Umschalter, SPDT-Wechselschalter (3 Anschlüsse, 2 stabile Zustände)
- Stufenschalter (n Anschlüsse, $n-1$ stabile Zustände)
- Taster (2 Anschlüsse, 1 stabiler Zustand)
  - Schließer (stabiler Zustand: geöffnet)
  - Öffner (stabiler Zustand: geschlossen)

Abbildung 1.5: Schaltzeichen für Einschalter, Umschalter, Schließer und Öffner

Die mechanische Bewegung für die Zustandsänderung kann Kippen, Drücken oder Drehen sein, seltener auch Ziehen oder Schieben. Abbildung 1.5 zeigt Schaltzeichen.

Widerstände

Ein Widerstand R ist ein elektronisches Bauteil, dessen wichtigste Eigenschaft durch die physikalische Größe Widerstand, Symbol R, beschrieben wird. Der Name „Widerstand“ steht also für zwei verschiedene Begriffe. Schaltzeichen für Festwiderstände, das sind Widerstände mit konstantem Sollwert, zeigt Abbildung 1.6.

Abbildung 1.6: europäisches und amerikanisches Schaltzeichen für einen Festwiderstand

Bei Festwiderständen überwiegen zwei Ausführungen: Kohleschichtwiderstände und Metallschichtwiderstände. Beide bestehen im Inneren aus einem Keramik-Zylinder. Außen ist entweder eine Schicht aus amorphem Kohlenstoff oder aus Metall, zum Beispiel Nickel-Chrom. Kohleschichtwiderstände sind etwas billiger, aber sie haben meistens eine höhere Toleranz. Das bedeutet, dass der tatsächliche Widerstandswert typischerweise um bis zu 5 % vom Sollwert abweichen kann. Metallschichtwiderstände haben typischerweise eine Toleranz von 1 %. Der Sollwert R_{Soll} und die Toleranz werden oft auf dem Bauteil R in Form von Farbringen angegeben.

Bei Kohleschichtwiderständen sind es in der Regel vier Ringe. Bei den ersten drei Ringen steht jede Farbe für eine Ziffer, siehe Tabelle 1.11.

schwarz	braun	rot	orange	gelb	grün	blau	violett	grau	weiß
0	1	2	3	4	5	6	7	8	9

Tabelle 1.11: Zuordnung von Farben zu Ziffern im Widerstands-Farbcode

Nennen wir die ersten drei Ziffern a, b und c, dann ist

$$R_{\text{Soll}} = (10 \cdot a + b) \cdot 10^c \tag{1.1}$$

Der vierte Ring gibt an, wie groß die Abweichung $|R - R_{\text{Soll}}| \;/\; R_{\text{Soll}}$ höchstens sein sollte. Ist er goldfarbig, beträgt die maximale Abweichung 5 %. Außerdem erkennt man, dass es der vierte Ring ist und nicht der erste, denn der erste kann nicht goldfarbig sein.

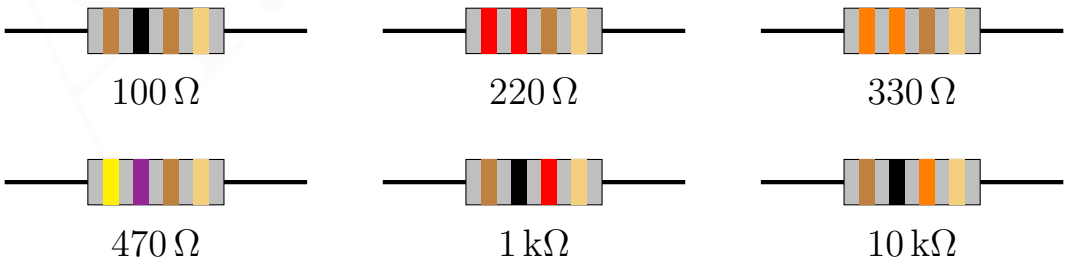


Abbildung 1.7: Beispiele für die Farbcodierung von Widerständen mit vier Ringen

Statt einer Kennzeichnung mit Farbringen können Widerstände eine Beschriftung im RKM-Code tragen. Wenn anstelle des Dezimalkommas für den Widerstandswert ein `R` steht, ist die Einheit Ω, mit `K` entsprechend kΩ und mit `M` ist es MΩ.

Man sollte den gewünschten Widerstand anhand des Farbcodes heraussuchen, aber vor der Verwendung des Widerstands seinen Wert mit einem Spannungsmessgerät oder einem Multimeter auch einmal messen.

Superkondensatoren

Ausgehend von Forschungen in den 1950er Jahren wurden sogenannte Superkondensatoren gegen Ende der !970er Jahren praktisch einsetzbar. Durch technische Fortschritte werden Superkondensatoren immer besser. Gegenüber Lithium-Ionen-Akkumulatoren haben sie Vor-und Nachteile. Langfristig könnten sie die Lithium-Ionen-Akkumulatoren zunehmend ersetzen.

Superkondensatoren haben im Vergleich zu herkömmlichen Kondensatoren eine viel größere Fläche. Da die Kapazität eines Kondensators proportional zur Fläche ist, wird eine viel größere Kapazität erreicht. Außerdem lagern sich bewegliche Ladungsträger dicht an die Elektroden und bilden eine Gegenelektrode mit sehr geringem Abstand. Da die Kapazität eines Kondensators umgekehrt proportional zum Abstand ist, ergibt sich eine besonders große Kapazität.

Die Energiedichte der Superkondensatoren beträgt nur ein Zehntel der von Lithium-Ionen-Akkus, aber sie ist für Anwendungen im Energy harvesting (Gewinnung kleiner elektrischer Energie für Geräte mit geringem Verbrauch) oft ausreichend.

Die Lebensdauer von Superkondensatoren ist, auch bei häufigem Entladen (Zyklenfestigkeit), länger als 10 Jahre. Nachteilig ist, dass die Ausgangsspannung während der Entladung sinkt. Außerdem können Superkondensatoren, abhängig vom Typ, eine hohe Selbstentladung haben. Ein Beispiel für Goldcap (Handelsname der Firma Panasonic) 1 F Superkondensatoren mit einer maximalemn Spannung von 5,5 V zeigt Abbildung 1.8.

Dioden

Dioden sind elektronische Bauteile mit zwei Anschlüssen (Zweipole), die eine nichtlineare Kennlinie haben. Das bedeutet: Trägt man den Strom durch eine Diode gegen den Spannungsabfall über der Diode auf, ergibt sich keine Gerade. Die statische Kennlinie ergibt sich für Gleichströme I. Sie verläuft durch (0 V,0 A).

In einem Intervall der Spannung U, das 0 V einschließt, gilt: Bei Spannungen U mit $I(U) \geq I(-U)$ spricht man von Durchlass, sonst von Sperrung. Die Vorzeichen von Spannung U und Strom I werden so gewählt, dass U und I in Durchlassrichtung positiv sind (Wahl der Zählpfeile).

Für ein kleines Intervall von U, das 0 V einschließt, mit kleinem Strom I, kann die statische Diodenkennlinie durch

$$I = I_{\mathrm{Rmax}} \cdot (e^{U\,/\,(n \cdot U_T)} - 1) \tag{1.2}$$

beschrieben werden. Dabei ist U_T die sogenannte Temperaturspannung ist (26 mV bei Raumtemperatur). I_{Rmax} ist eine spannungsunabhängige Größe, die vom Diodentyp,

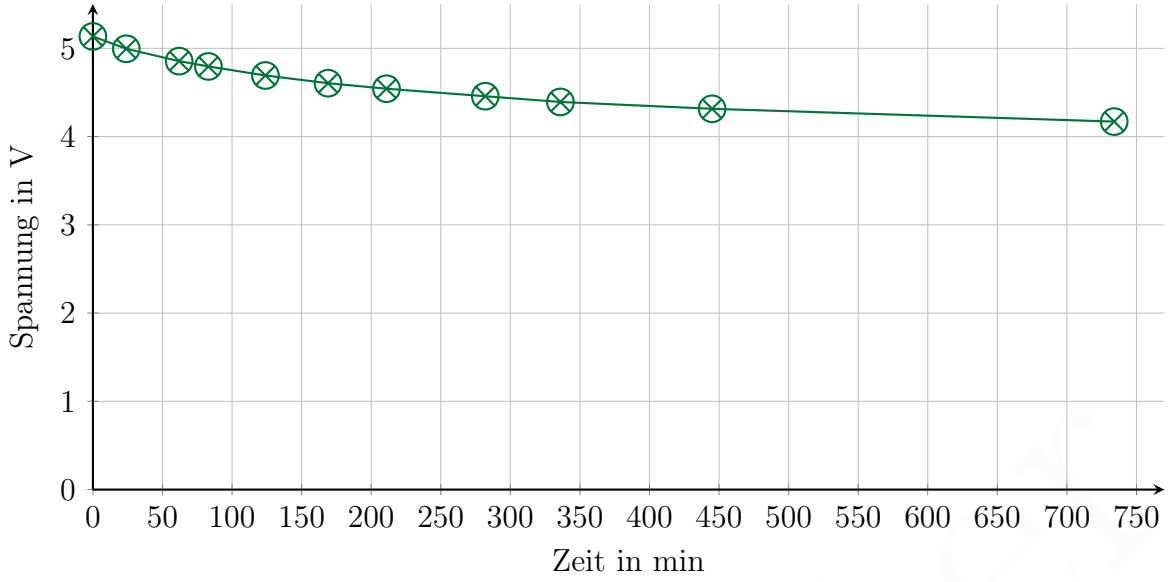


Abbildung 1.8: Selbstentladung billiger 1 F Superkondensatoren (Mittelwerte, n = 2)

insbesondere vom Material abhängt. Für Siliziumdioden ist $I_{\mathrm{Rmax}} \approx 10\,\mathrm{nA}$ und für Germaniumdioden ist $I_{\mathrm{Rmax}} \approx 10\,\mu\mathrm{A}$. n heißt Emissionskoeffizient oder Idealitätsfaktor. Bei einer idealen Diode ist $n = 1$, bei realen Dioden liegt n meistens zwischen 1 und 2. Für Dioden des Typs 1N4148 ist $n \approx 1{,}9$.

Derjenige Anschluss einer Diode, der bei Durchlass Pluspol ist, heißt Anode; der andere Kathode. Dies gilt unabhängig von Definitionen für „Anode“ und „Kathode“ in anderen Zusammenhängen. Als Bauteil hat eine Diode (im Neuzustand) oft zwei unterschiedlich lange Anschlussdrähte. Der kürzere ist dann die Kathode, der längere die Anode.

Aufgrund der nichtlinearen Kennlinie können Dioden zur Gleichrichtung von Wechselströmen eingesetzt werden. Im Hinblick auf die Einsatzgebiete unterscheidet man Gleichrichterdioden und Schaltdioden. Tabelle 1.12 listet einige Typen auf. Gleichrichterdioden dienen der Gleichrichtung von Wechselströmen mit großer elektrischer Leistung meistens niedriger Frequenz, insbesondere Netzfrequenz (50 Hz oder 60 Hz). Schaltdioden, auch Signaldioden oder Kleinsignaldioden genannt, sollen dagegen relativ kleine Signale schnell verarbeiten.

Sondertypen sind Z-Dioden (auch Zener-Dioden genannt), Leuchtdioden und Kapazitätsdioden. Leuchtdioden (kurz LED) sind Dioden, die in Vorwärtsrichtung einen Teil der in der Diode verbrauchten elektrischen Leistung (Durchlassverlustleistung P_{F}) in Licht (statt Wärme) wandeln. Schaltzeichen für Dioden zeigt Abbildung 1.9.

Z-Dioden

Wir gehen zunächst von der allgemeinen Definition von Spannung U und Strom I für Dioden aus (siehe oben). Werden Dioden in Sperrrichtung betrieben ($U < 0\,\mathrm{V}, I < 0\,\mathrm{A}$)),

1N4148	Schaltdiode bis 100 MHz, $U_\mathrm{F}(0{,}1\,\mathrm{mA}) = 0{,}50\,\mathrm{V}$, $U_\mathrm{F}(1\,\mathrm{mA}) = 0{,}62\,\mathrm{V}$
1N4448	Schnelle Kleinsignal-Schaltdiode, I_F max. 300 mA, $I_\mathrm{R}(20\,\mathrm{V}) \leq 25\,\mathrm{nA}$
1N5406	Gleichrichterdiode, bis 3 A Vorwärtsdauerstrom an ohmscher Last
BAT 41	Schottky-Schaltdiode, $U_\mathrm{F}(0{,}1\,\mathrm{mA}) = 0{,}3\,\mathrm{V}$, $U_\mathrm{F}(1\,\mathrm{mA}) = 0{,}4\,\mathrm{V}$
BAT 43	Schottky-Schaltdiode, $U_\mathrm{F}(0{,}1\,\mathrm{mA}) = 0{,}2\,\mathrm{V}$, $U_\mathrm{F}(1\,\mathrm{mA}) = 0{,}3\,\mathrm{V}$
BAT 86	Schottky-Schaltdiode, $U_\mathrm{F}(0{,}1\,\mathrm{mA}) = 0{,}17\,\mathrm{V}$, $U_\mathrm{F}(1\,\mathrm{mA}) = 0{,}24\,\mathrm{V}$
BY500-800	Gleichrichterdiode, I_F max. 5 A, $I_\mathrm{R}(800\,\mathrm{V}) \leq 10\,\mu\mathrm{A}$, $U_\mathrm{F}(5\,\mathrm{A}) \leq 1{,}3\,\mathrm{V}$

Tabelle 1.12: Beispiele und Daten von Dioden

Abbildung 1.9: Schaltzeichen für verschiedene Dioden. Die Anode ist links.

steigt $|I|$ mit $|U|$ an. In diesem Bereich arbeiten Z-Dioden. Der maximal erlaubte Betrag des Stroms ist I_Zmax. Bei größerem $|I|$ wird die Diode zerstört.

Das wichtigste Merkmal einer Z-Diode ist die Z-Spannung U_Z. Es gibt zwei verschiedene Definitionen:

a)

$$|I| = \frac{I_\mathrm{Zmax}}{10} \quad \Longrightarrow \quad U_\mathrm{Z} = |U| \tag{1.3}$$

b)

$$I = c_1 \cdot e^{(U+U_\mathrm{Z})\,/\,c_2} \quad \Longrightarrow \quad U_\mathrm{Z} = |U| \tag{1.4}$$

wobei $c_1 < 0$ und $c_2 < 0$ Konstanten sind.

Beide Definitionen führen in der Praxis zu ähnlichen Werten. Z-Dioden werden für bestimmte U_Z-Werte angeboten, zum Beispiel 4,7 V, 5,1 V, 5,6 V, 6,2 V, 6,8 V, 7,5 V, 8,2 V, 9,1 V, 8,2 V, 10 V und , 11 V.

Da $I(U)$ eine exponentielle Funktion ist, gibt es in der Kennlinie keinen mathematischen „Knick“ (Stelle ohne Differenzierbarkeit). Allerdings: Bei einer bestimmten (sinnvollen, da in der Praxis oft nützlichen) Skalierung der Kennlinie scheint es einen „Knick“ zu geben. Aber das ist ein graphischer Effekt, der auf einer Skalierung mit der Beschränkung durch $U(I_\mathrm{Zmax})$ beruht.

Der dynamische Widerstand $r_\mathrm{Z} = {}^{\mathrm{d}U}/_{\mathrm{d}I}$ hängt von U_Z und I ab. Bei einem Strom von $I = -5\,\mathrm{mA}$ oder $I = -10\,\mathrm{mA}$ ist r_Z für $6{,}2\,\mathrm{V} \leq U_\mathrm{Z} \leq 8{,}2\,\mathrm{V}$ besonders niedrig. Eine sehr niedrige Temperaturabhängigkeit hat man bei $U_\mathrm{Z} = 5{,}1\,\mathrm{V}$ und $U_\mathrm{Z} = 5{,}6\,\mathrm{V}$.

Leuchtdioden (LED)

Leuchtdioden (englisch Light Emitting Diode, kurz LED) wandeln elektrische Energie in Licht, wenn sie von einem Strom durchflossen werden. Sie leuchten nur in Vorwärtsrichtung des Stroms (Durchlass). Anders herum, in Sperrrichtung, bleiben sie dunkel.

In einer typischen Signal-LED für ergibt sich bei einem Vorwärtsstrom von 10 mA eine Durchlass-Spannung $u_\mathrm{F}(10\,\mathrm{mA})$, die vom Halbleitermaterial, und damit indirekt von der Lichtfarbe abhängt. $u_\mathrm{F}(10\,\mathrm{mA})$ liegt zwischen 1,6 V (rot) und 3 V (blau). Bei 2 V ergibt sich $P_\mathrm{F} = 20\,\mathrm{mW}$.

Eine Standard-LED, siehe Abbildung 1.10 links, hat eine Anode (A) und eine Kathode (K). Die Anode, erkennbar am längeren Anschlussdraht, muss zum positiven Potential zeigen, die Kathode zum negativen. Das Schaltzeichen zeigt Abbildung 1.9.

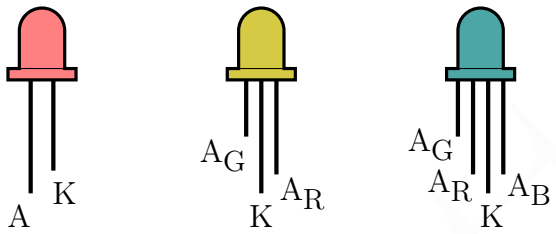


Abbildung 1.10: Anschlüsse von einer Standard-LED, einer Duo-LED mit gemeinsamer Kathode und einer Dreifach-LED mit gemeinsamer Kathode

Eine Signal-LED wird mit meistens mit einem Vorwiderstand betrieben, der den Strom begrenzt. Alternativ kann die LED an eine Konstantstromquelle angeschlossen werden.

Das Gehäuse einer einfachen Standard-LED ist an der Kathodenseite abgeflacht. Bei einer neuen einfachen Standard-LED ist der Anschlussdraht an der Anode länger als an der Kathode. Die Anschlussdrähte bestehen aus verzinntem Stahl, dessen Wärmeleitfähigkeit geringer ist als bei Kupfer. Damit kann eine LED ohne Überhitzung eingelötet werden.

Statt einer einfachen LED kann man eine zweifarbige Duo-LED verwenden. Sie besteht aus zwei, in einem Gehäuse zusammengefassten, LED-Teilen verschiedener Farbe, zum Beispiel rot und grün. Eine solche Duo-LED kann rot, grün oder gelb leuchten; letzteres wenn rot und grün gleichzeitig leuchten. Die Ansteuerung erfolgt meist über drei Anschlüsse, mit gemeinsamer Kathode oder gemeinsamer Anode. Abbildung 1.10 zeigt in der Mitte eine Duo-LED mit gemeinsamer Kathode (K) und je einer Anode für grün ($\mathrm{A_G}$) und rot ($\mathrm{A_R}$). Anders als bei der einfachen Standard-LED, kann der Kathodendraht länger als die Anodendrähte sein. Für beide LED-Teile wird ein Vorwiderstand benötigt, je einer vor $\mathrm{A_G}$ und vor $\mathrm{A_R}$. Die beiden werden wie zwei einzelne LED an- und ausgeschaltet.

Entsprechend enthält eine Dreifach-LED drei LED-Teile verschiedener Farbe, zum Beispiel rot, grün und blau. Abbildung 1.10 zeigt rechts eine Dreifach-LED mit gemeinsamer Kathode. Eine derartige Dreifach-LED ist zum Beispiel der Typ `LL-509RGBC2E-006` der Firma Luckylight Electronics (Shenzhen, China).

LED, die als Leuchtmittel dienen, sollen weißes Licht aussenden. Eine Parallelschaltung von roten, grünen und blauen LED kann zwar einen weißen Lichteindruck erzeugen, aber die Wiedergabe von Körperfarben ist schlecht, weil das Spektrum des Lichts nur drei Wellenlängenintervalle enthält, zwischen denen Lücken sind.

Für weiß leuchtende LED verwendet man daher meistens eine kurzwellige (blau-violette) LED, deren Licht mit einem Farbstoff durch Photolumineszenz (Fluoreszenz) größtenteils in ein breites Lichtspektrum gewandelt wird, mit einem lokalen Maximum im mittleren Wellenlängenbereich (gelb). Allerdings bleibt auch ein lokales Maximum im blauen Bereich und darum erscheint das Licht insgesamt als weiß.

Da eine weiß leuchtende LED eine blau-violette LED enthält, liegt ihre Durchlass-Spannung u_F für typische Dauervorwärtsströme ebenfalls bei rund 3 V, und bei gepulsten Strömen etwas höher (Pulsweitenmodulation zur Leistungsregelung, Dimmen genannt). Allerdings steigt u_F mit dem Vorwärsstrom an, siehe Beispiel in Tabelle 1.13.

I / mA	1	2	5	10	20	30	40	50
u_F / V	2,54	2,59	2,68	2,78	2,94	3,06	3,17	3,24
P / mW	2,54	5,18	13,4	27,8	58,8	91,8	126,8	162

Tabelle 1.13: Dauerstrom I, Spannung u_F und Leistung P einer weißen LED des Typs SLOAN L5-N52N-FTU (Sibalco AG, Basel) bei 20 °C; der Dauerstrom sollte 30 mA nicht, oder nur kurzzeitig übersteigen (Messung vom 02.06.2023)

Eine Reihenschaltung von zwei weißen LED des Typs SLOAN L5-N52N-FTU (Sibalco AG, Basel), an der eine Gleichspannung von 5 V anlag, ergab bei 20 °C einen Vorwärtsstrom von 0,36 mA. Das entspricht einer Leistung von 0,9 mW pro LED. Das genügt um die LED als Signalleuchten zu verwenden. Die LED können auch als schwaches Orientierungslicht in völliger Dunkelheit dienen. Wenn man die LED ohne Vorwiderstand mit der 5 V Spannung des Raspberry Pi speist, sollte man eine Feinsicherung in Reihe mit den LED schalten.

Vorteil dieser Schaltung ist die optimale Energieeffizienz, da es keine Verlustleistung an Vorwiderständen oder aktiven Bauelementen gibt. Außerdem ist das Verhältnis von Leuchtkraft und elektrischer Leistungsaufnahme einer LED näherungsweise konstant und bei genauerer Berechnung steigt es sogar, wenn die elektrische Leistung verringert wird. Abgesehen von den Kosten der Schaltung, ist der Betrieb mehrerer LED mit gleicher Gesamtleuchtkraft also effizienter als der Betrieb einer einzelnen LED.

Die optische Achse einer LED zeigt in Richtung maximaler Lichtstärke (englisch: luminous intensity). Für einen festen Abstand von der LED gibt es einen Kegel (mit der LED in der Spitze), dessen Mantel einen Öffnungswinkel hat, bei dem die Lichtstärke halbmaximal wird. Dieser Öffnungswinkel heißt Abstrahlwinkel oder Halbwertswinkel (englisch: beam angle[8]). Entsprechen heißt der Öffnungswinkel, bei dem die Lichtstärke auf $^1/_{10}$ abgefallen ist, Feldwinkel (field angle).

[8] Statt beam angle findet man auch die englische Bezeichnung viewing angle. Manche Hersteller von LED bezeichnen leider den halben beam angle als viewing angle.

Transistoren

Wir betrachten nur Bipolartransistoren. Ein Transistor hat drei Anschlüsse: Basis (B), Collektor (C) und Emitter (E), siehe Abbildung 1.11.

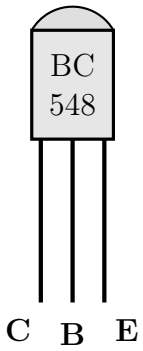


Abbildung 1.11: Anschlüsse des Transistors BC 548: **C**ollektor, **B**asis und **E**mitter

Es gibt zwei Sorten: npn (Basis p-dotiert, Emitter und Collektor n-dotiert) und pnp (Basis n-dotiert, Emitter und Collektor p-dotiert). Abbildung 1.12 zeigt Schaltzeichen.

Abbildung 1.12: Schaltzeichen für bipolare Transistoren: npn und pnp

Transistoren sind mit einer Zeichenkette beschriftet. Ein häufiges Kennzeichnungsschema beginnt mit zwei Buchstaben, gefolgt von einer dreistelligen Zahl. Der erste Buchstabe bezeichnet das Halbleitermaterial (A = Germanium, B = Silizium), der zweite Buchstabe gibt einen Hinweis auf die bevorzugte Anwendungsart (C = NF-Transistor, D = NF-Leistungstransistor, F = HF-Transistor, L = HF-Leistungstransistor, S oder U = Schalttransistor). Die Zahl gibt die Baureihe an. Danach folgt oft ein weiterer Buchstabe (A, B oder C), der den Typ anhand einer Kenngröße, der Stromverstärkung, genauer angibt.

Beispiele: BC548B ist ein npn-Transistor aus Silizium für niederfrequente Signale kleiner Leistung mit einer Stromverstärkung zwischen 200 und 450. BC558B ist ein pnp-Transistor mit ansonsten ähnlichen Eigenschaften.

Die Kleinsignal-npn-Transistortypen BC546, BC547, BC548 und BC549 haben ähnliche Eigenschaften und sind in vielen Schaltungen austauschbar. Der Grundtyp BC548 hat einen größeren Bereich an Stromverstärkungen, während BC547 mit etwas höheren Spannungen betrieben werden kann (45 V statt 30 V), aber mehr Rauschen hat. BC549 weist sehr geringeres Rauschen auf. BC546 ist für besonders hohe Spannungen ausgelegt.

BC550 verbindet hohe Spannung mit geringem Rauschen. Wenn ein Stromverstärkungstyp angegeben wird, bedeutet A niedrige (110 – 220), B mittlere (200 – 450) und C hohe (420 – 800, oder – 900). Genauere Angaben findet man natürlich in den Datenblättern.

Temperatursensor DS18S20

Der Temperatursensor DS18S20 der Firma Maxim/Dallas (ein Nachfolgemodell des Temperatursensors DS1820) wandelt Temperatur zunächst in eine analoge elektrische Spannung und dann, mithilfe eines AD-Wandlers, in digitale Signale, die über einen Eindraht-Bus (1-Wire) 1 gesendet werden. Da der Sensor auf einer Steckplatine aufgebaut werden soll, wird die Bauform TO-92 mit drei Anschlussbeine (Pins) benutzt. Die Anschlüsse sind GND, DQ und V_{DD}, siehe Abbildung 1.13. Der TO-92-DS18S20 ist von Elektronikbauteilehändlern leicht erhältlich und das zugehörige Datenblatt findet man im Internet.

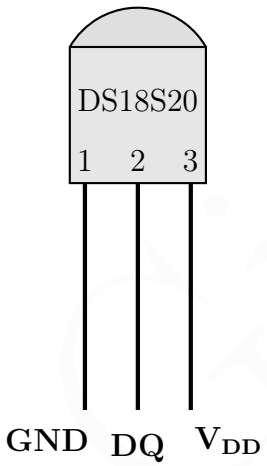


Abbildung 1.13: Anschlüsse des Temperatursensors DS18S20 im TO-92 Gehäuse: Pin 1 = GND (Masse), Pin 2 = DQ (Daten) und Pin 3 = V_{DD} (Power, 3,3 V).

Temperatursensor LM75

LM75 ist ein Temperatursensor der Firma Texas Instruments, welche die Firma National Semiconductor übernahm. Er wandelt (ähnlich wie der DS18S20, siehe oben, Seite 30) Temperatur zunächst in eine analoge elektrische Spannung und dann mithilfe eines AD-Wandlers in digitale Signale, die über einen seriellen Datenbus zum Raspberry Pi gesendet werden. Der LM75 ist von Elektronikbauteilehändlern leicht erhältlich und das zugehörige Datenblatt findet man im Internet.

Der LM75 wird in der SO Bauform als SMD geliefert. Die acht Anschlussbeinchen (Pins) sind kurz, eng benachbart und brechen leicht ab. Daher ist es schwierig, ihn direkt mit Anschlussleitungen zu versehen oder ihn mit einer Steckplatine zu verbinden. Am einfachsten ist es, den LM75 auf eine SO-8 Adapterplatine zu löten. Mit Geschick kann man den LM75 auch auf eine normale Lochrasterplatine (oder eine andere Leiterplatte) kleben und Anschlussleitungen zugentlastet (über Lötaugen) zum Raspberry Pi führen.

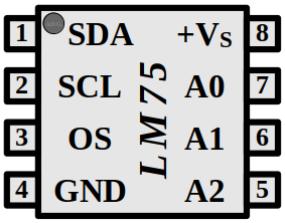


Abbildung 1.14: Anschlüsse des LM75 im SO-8 Gehäuse: Pin 1 = SDA, Pin 2 = SCL, Pin 3 = OS (Overtemperature Shutdown) benutzen wir nicht, Pin 4 = GND (Masse), Pins 5, 6 und 7 = A2, A1, A0 (Adressenfestlegung für den Bus, I^2C address selection), Pin 8 = $+\,V_S$ (supply voltage 3,3 V).

Man beachte, dass es zwei Ausführungen des LM75 gibt, die entweder für eine Versorgungsspannung (supply voltage) von 3,3 V oder von 5 V ausgelegt sind. Da die Signalleitungen für den I^2C Bus des Raspberry Pi mit einer Spannung von 3,3 V arbeiten und wir keinen Spannungswandler einsetzen wollen, müssen wir darauf achten, dass wir eine Ausführung des LM75 benutzen, die auf eine Versorgungsspannung von 3,3 V ausgelegt ist. Manche Ausführungen des LM75 für eine Versorgungsspannung von 5 V funktionieren auch mit einer Versorgungsspannung von 3,3 V, aber mit geringerer Auflösung.

Ein Schaltplan für den Anschluss des LM75 wird ab Seite 36 vorgestellt.

A/D-Wandler MCP3208

Der MCP3208 ist ein Analog/Digitalwandler, der analoge Messdaten aufnimmt und digital über einen Datenbus weitergibt. Wichtige Eigenschaften sind

- Auflösung 12 bit
- 8 Eingangskanäle (single-ended: Signal gegen Masse)
- alternativ 4 differentielle Eingangskanäle (Eingangspaare)
- Datenbus: Serial Peripheral Interface (SPI)
- eine Versorgungsspannung V_{DD} zwischen 2,7 V und 5,5 V
- maximale Abtastrate 50 000 s^{-1} bei V_{DD} = 2,7 V (100 000 s^{-1} bei V_{DD} = 5 V)
- mögliches Gehäuse: Plastic Dual In-line (P), 16 Pins

Der MCP3208 hat 16 Anschlüsse (Pins), siehe Abbildung 1.15. Pins 1 bis 8 stellen die Eingänge für die zu messenden Analogspannungen dar und werden mit CH0 bis CH7 bezeichnet. Die analogen Eingangsspannungen beziehen sich auf Pin 14 des MCP3208 (AGND, analog ground). Programmgesteuert können sie entweder 8 verschiedene unipolare Spannungen (gegen analoge Signalmasse) messen oder 4 verschiedene bipolare Spannungen (differentiell, CH0 – CH1, CH2 – CH3, usw.). Pin 9 (V_{DD}) liefert die Versorgungsspannung V_{DD} für den MCP3208. Sie muss zwischen 2,7 V und 5,5 V liegen. Beim Anschluss an den Raspberry Pi ist es am einfachsten, die Spannung der GPIO (3,3 V) als Versorgungsspannung zu nehmen. Pin 9 (DGND) bildet die digitale Signalmasse. Wir werden sie mit der Signalmasse des Raspberry Pi verbinden.

Der Datenbus SPI belegt Pins 10 bis 13. Pin 10 ($\overline{CS}$) wählt den MCP3208 im Datenbus

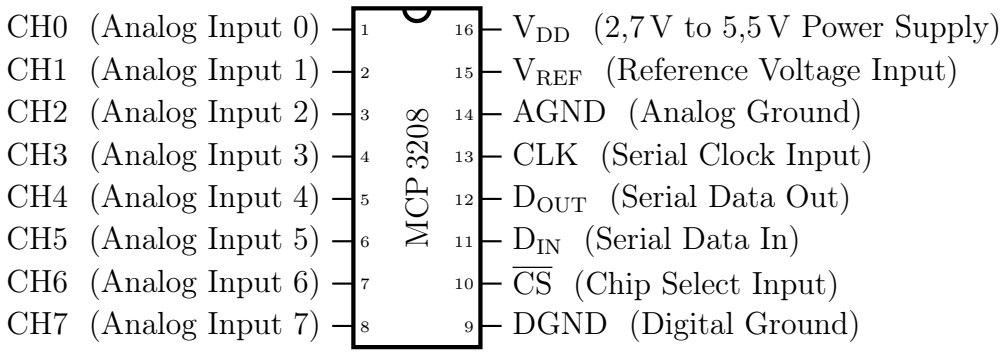


Abbildung 1.15: Anschlüsse des A/D-Wandlers MCP 3208

aus, wenn $\overline{\text{CS}}$ *low* ist (umgekehrte Logik, daher der Überstrich). Pin 11 (D_{IN}) empfängt Steuerdaten (channel configuration data). Pin 12 (D_{OUT}) sendet die digitalen Daten. Pin 13 (CLK) empfängt den Taktgeber (clock, SCLK Signal) des Datenbusses.

Pin 14 (AGND) bildet die Signalmasse für die analogen Schaltungsteile des MCP3208. In vielen Anwendungen, aber nicht immer, wird sie mit der digitalen Signalmasse verbunden. Pin 15 (V_{REF}) bestimmt den Messbereich der Analogeingänge. Ist die Spannung an einem Analogeingang gleich V_{REF}, wird der maximale digitale Wert 4096 (`111111111111`) erzeugt. Pin 16 (V_{DD}) nimmt die Versorgungsspannung V_{DD} entgegen, die zwischen 2,7 V und 5,5 V liegen muss. Beim Anschluss an den Raspberry Pi ist es am einfachsten, die Spannung der GPIO, 3,3 V, als Versorgungsspannung zu nehmen.

Um Störungen und Rauschen der Versorgungspannung V_{DD} zu minimieren, sollte zwischen Pins 14 (AGND) und 16 (V_{DD}) des MCP3208 ein Abblockkondensator von 1 µF gesetzt werden. Er wird möglichst nah an den Pins platziert. Geeignet sind insbesondere Keramikkondensatoren, während zum Beispiel gewickelte Kondensatoren wegen ihrer Induktivität weniger geeignet sind.

Varianten des MCP3208 sind der MCP3201, der MCP3202 und der MCP3204 mit 1, 2 beziehungsweise 4 Eingängen für Analogspannungen. Den MCP3208 und seine Varianten gibt es in einer B-Version und einer weniger genauen C-Version.

1.2.4 Externe Schaltungen

Schaltung: Einfacher GPIO-Anschluss einer einfachen LED

Mit einem GPIO (zum Beispiel GPIO 23) kann man eine Leuchtdiode an- und ausschalten. Der Leuchtdiodenschaltkreis besteht aus der Reihenschaltung einer Leuchtdiode (Standard-LED) und eines passenden Vorwiderstands R (zum Beispiel 330 Ω oder 470 Ω). Die Anode der Leuchtdiode muss in Richtung GPIO 23 zeigen (Pin 16 der Stiftleiste J8 beziehungsweise P1); die Kathode in Richtung GND (Pin 9 von J8). Die Reihenfolge von Widerstand und LED ist beliebig, siehe Abbildung 1.16.

Der Leuchtdiodenschaltkreis kann auf einer Steckplatine aufgebaut werden. Mit einem Computerprogramm kann GPIO 23 als Datenausgang geschaltet werden.

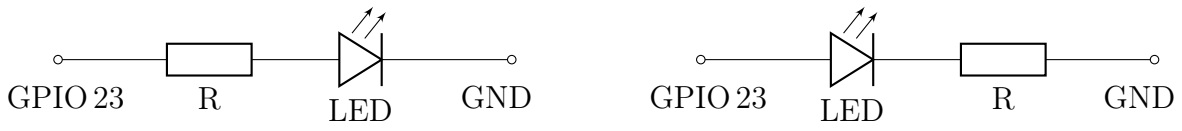


Abbildung 1.16: Verschaltung einer LED, die Reihenfolge von R und LED ist beliebig

In der Regel gilt: Wenn das Programm eine logische 0 ausgibt, liegt 0 V an GPIO 23; wenn es eine logische 1 ausgibt, liegt 3,3 V an GPIO 23 und die Leuchtdiode leuchtet. Es ist aber möglich, eine umgekehrte Logik zu verwenden, sodass eine logische 0 3,3 V und eine logische 1 0 V ausgibt.

Schaltung: Dreifacher GPIO-Anschluss einer Dreifach-LED

Eine Dreifach-LED wird über je drei GPIO und Vorwiderstände angesteuert. Die Schaltung für eine Dreifach-LED mit gemeinsamer Kathode (siehe Abbildung 1.10 auf Seite 27, rechts) wird in Abbildung 1.17 gezeigt.

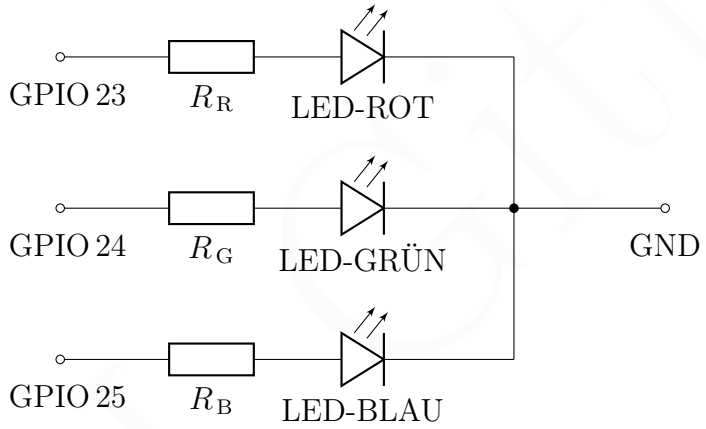


Abbildung 1.17: Verschaltung einer Dreifach-LED mit gemeinsamer Kathode

Für den Typ `LL-509RGBC2E-006` schlägt Tabelle 1.14 Vorwiderstände vor.

rote LED	grüne LED	blaue LED
GPIO 23 $R_R = 470\,\Omega$	GPIO 24 $R_G = 220\,\Omega$	GPIO 25 $R_B = 150\,\Omega$

Tabelle 1.14: Beispiel für die Beschaltung einer Dreifach-LED

Schaltung: Anschluss einer LED über einen Transistor

Der Laststrom, den die GPIO liefern können, ist (wie oben beschrieben) beschränkt und daher ist es besser, sie nur zum Steuern (Ein- und Ausschalten) zu benutzen. Die

elektrische Leistung für die Last (hier: eine LED) sollte von der Energieversorgung abgenommen werden. Abbildung 1.18 zeigt eine entsprechende Schaltung.

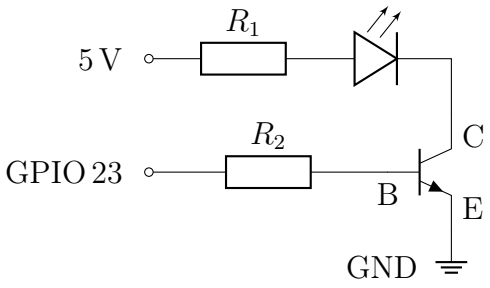


Abbildung 1.18: LED-Ansteuerung über einen npn-Transistor mit Emitterschaltung

Als elektronische Bauteile werden eine Standard-LED, ein npn-Transistor und zwei Widerstände (330 Ω, 10 kΩ) eingesetzt.

Wir verwenden einen Transistor vom Typ BC547 oder einen anderen `npn` Kleinsignaltransistor. Es gibt ihn in unterschiedlichen Subtypen (BC547A, BC547B und BC547C), die sich in der Stromverstärkung unterscheiden. Da es keinen nennenswerten Preisunterschied bei den Subtypen gibt und hier, wie meistens, eine höhere Stromverstärkung vorteilhaft ist, wählen wir (sofern wir die Wahl haben) den Subtyp BC547C mit der höchsten Stromverstärkung. Statt mit BC547C kann der Transistor auch (kürzer) mit C547C beschriftet sein. Schaut man auf die flache (beschriftete) Seite eines BC547 im Kunststoffgehäuse TO-92, so ist der linke Anschluss der Collektor C, rechts der Emitter E und in der Mitte die Basis B, siehe Abbildung 1.11.

Um den Transistor in eine Platine einstecken zu können, wählen wir eine Ausführung im Kunststoffgehäuse TO-92 und nicht im SMD Gehäuse. Der BC547 besteht aus Silizium (erkennbar am ersten Buchstaben B der Bezeichnung) und wird meistens bei Signalen kleiner Leistung und niedriger Frequenzen verwendet (erkennbar am zweiten Buchstaben C der Bezeichnung).

Die Werte der beiden Widerstände sollten ungefähr $R_1 = 330\,\Omega$ und $R_2 = 10\,\mathrm{k}\Omega$ betragen. Ein Multimeter zur Messung der Spannungen und Ströme in der Schaltung ist wünschenswert. Am einfachsten baut man die Schaltung auf einer Steckplatine mit passenden Kabeln und Drahtbrücken auf. Man kann sie auch auf eine Platine löten.

GPIO 23 liefert im Zustand *high* (logische 1 = 3,3 V) einen kleinen Steuerstrom I_{BE} zur Basis B des Transistors, welcher zur Signalmasse am Emitter E abfließt. Damit wird ein großer Laststrom I_{CE} zwischen Collector C und Emitter E eingeschaltet, der von der 5 V Spannung der Energieversorgung gespeist wird.

Ausgehend von einem Transistor des Typs BC547C wollen wir mithilfe des Datenblatts sicherstellen, dass der Transistor in unserer Schaltung nicht überfordert wird: Im Datenblatt finden wir unter absolute maximum ratings ($T_{\mathrm{a}} = 25\,\mathrm{degreeCelsius}$) die maximal zulässige Spannung zwischen Collektor und Emitter (collector emitter voltage) V_{CE} von 45 V. T_{a} steht für ambient temperature, zu Deutsch Umgebungstemperatur. Für diese gelten die Angaben im Datenblatt. Da der Transistor nur eine Spannung von 5 V schaltet, ist $V_{\mathrm{CE}} \leq 5\,\mathrm{V}$ und damit deutlich unter der maximal zulässigen V_{CE}.

Der maximal zulässige fortdauernde (nicht nur kurzzeitg anliegende) Kollektorstrom (continuous collector current) I_C beträgt 100 mA. Dies liegt deutlich unter den höchstens 20 mA, mit denen wir eine LED im Collektorstromkreis betreiben. Beim vollständig leitenden (gesättigten) Transistor beträgt, mit $I_C = 20\,\text{mA}$, der Spannungsabfall $V_{CE}\,(\text{sat})$ zwischen Collektor und Emitter (collector emitter saturation voltage) 0,3 V oder weniger. Das ergibt eine (den Transistor erwärmende) Verlustleistung von höchstens $V_{CE}\,(\text{sat}) \cdot I_C = 0{,}3\,\text{V} \cdot 20\,\text{mA} = 6\,\text{mW}$. Da dies der Hauptbeitrag zur gesamten Verlustleistung des Transistors (power dissipation) ist, bleibt man deutlich unter der maximal zulässigen Verlustleistung von 500 mW.

Schaltung: Anschluss eines Tasters (oder Schalters)

Die Stellung eines Tasters oder Schalters kann über die GPIO bestimmt werden und so als digitale Eingabe des nutzers verwendet werden. In Abbildung 1.19 steuert ein Taster die Spannung $U_{\text{GPIO}\,22}$ an GPIO 22 (J8-Pin 15). Im offenen Zustand des Tasters liegt $U_{\text{GPIO}\,22} = 0\,\text{V}$ an, im geschlossenen $U_{\text{GPIO}\,22} = 3{,}3\,\text{V}$. Den Spannungen ordnen wir die logischen Zustände `0` und `1` zu.

Der GPIO muss auf die Betriebsart (Modus) *Eingabe* gesetzt werden (siehe unten).

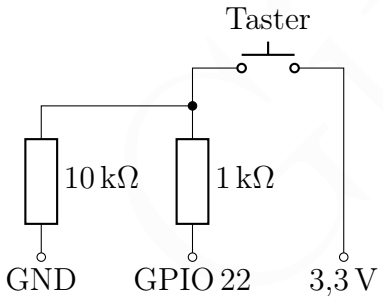


Abbildung 1.19: Verschaltung des Tasters

Schaltung: Temperaturmessung mit DS18S20

Der Temperatursensor DS18S20 (siehe Seite 30) wird über einen Eindraht-Bus (siehe Seite 20) an den Raspberry Pi angeschlossen. Abbildung 1.20 zeigt eine Schaltung zur Temperaturmessung mit dem DS18S20.

Über Pin 2 des DS18S20 werden binäre Daten empfangen oder gesendet. Die beiden logischen Zustände werden hier durch die Spannungen 0 V (logisch LOW) und 3,3 V (logisch HIGH) vermittelt. Ein 4,7 kΩ Widerstand wird zwischen Pin 2 und Pin 3 des TO-92-DS18S20 gesteckt.

Der Transistorausgang von Pin 2 des DS18S20 hat eine Open-Drain-Schaltung (so genannt bei Feldeffekttransistoren, bei Bipolartransistoren entspricht dies einer Open-Collector-Schaltung) und nimmt daher die Zustände 0 V oder Offen an. Offen bedeutet, dass der Transistor intern idealerweise vom Ausgang getrennt ist, real über einen großen

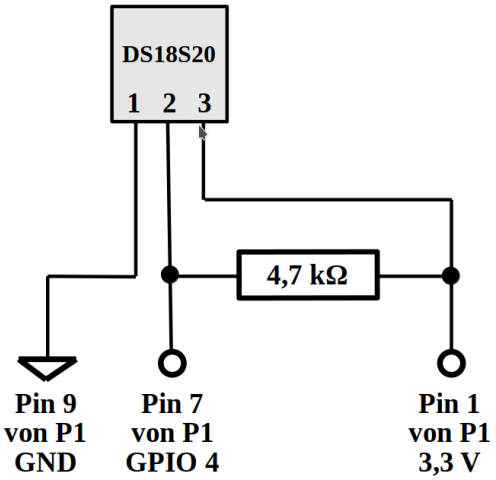


Abbildung 1.20: Temperaturmessung mit DS18S20. Pin 1 des DS18S20 wird mit GND des Raspberry Pi verbunden, hier mit Pin 9 der Stiftleiste J8 (oder P1 bei den alten Modellen 1 A und 1 B). Pin 2 des DS18S20 wird mit GPIO 4 (Pin 7 der Stiftliste) und Pin 3 des DS18S20 mit einer 3,3 V Spannungsquelle (hier Pin 1 der Stiftleiste).

Widerstand (hochohmig). Um im Offen-Zustand die Spannung 3,3 V bereitzustellen, wird der Ausgang über einen sogenannten Pullup-Widerstand mit der 3,3 V Versorgungsspannung verbunden. Wenn der Transistor durchschaltet, also der Ausgang auf 0 V gesetzt wird, fließt Strom durch den Widerstand; aber da der Transistorausgang niederohmig mit 0 V (Masse) verbunden ist, bleibt die Spannung nahe 0 V.

Pythonprogramme zur Temperaturmessung mit dieser Schaltung findet man im Abschnitt „Temperatursensor am 1-Wire Bus“ ab Seite 366.

Schaltung: Temperaturmessung mit LM75

Der Anschluss eines Temperatursensors LM75 für eine Versorgungsspannung (supply voltage) von 3,3 V (siehe Seite 30) an den I^2C Datenbus eines Raspberry Pi kann ohne weitere elektronische Bauteile erfolgen, siehe Abbildung 1.21 links. Pins 7 (A0), 6 (A1) und 5 (A2) legen die Busadresse des LM75 fest. Wenn wir diese Pins mit Masse (GND) verbunden haben, ist die Adresse 0x48.

Soll dem LM75 eine bestimmte Adresse auf dem I^2C Datenbus zugeordnet werden, verwendet man die in Abbildung 1.21 rechts gezeigte Schaltung.

Vier der LM75 Pins (SDA, SCL, GND, $+V_S$) führen zu Pins der Stiftleiste J8 des Raspberry Pi. Drei Pins des LM75 (A0, A1 und A2) werden im einfachsten Fall direkt mit Masse (GND) verbunden; der achte Pin (OS) wird hier nicht verwendet. Der Raspberry Pi stellt für den I^2C Bus die GPIO 2 und 3 bereit, welche (im Gegensatz zu anderen GPIO) mit einem internen 1,8 kΩ Pullup-Widerstand versehen sind, um die Beschaltung des I^2C Busses zu vereinfachen. Die Verbindung von LM75 und Raspberry Pi erfolgt im einfachsten Fall gemäß Abbildung 1.21 links und Tabelle 1.15.

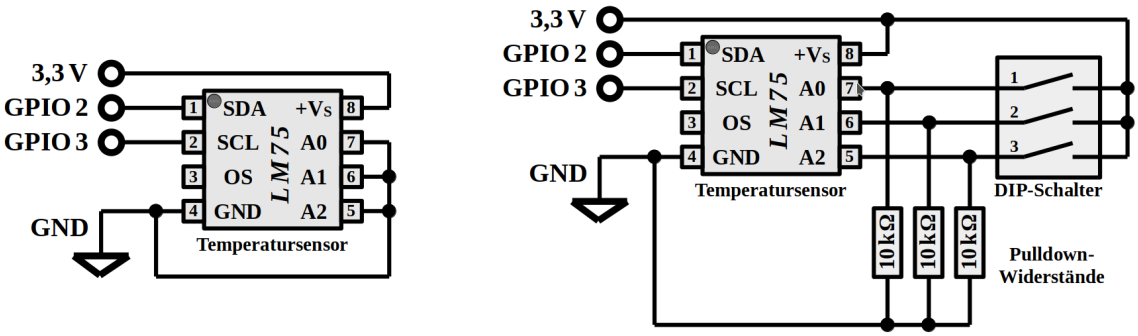


Abbildung 1.21: Schaltpläne für den Anschluss des LM75. Links: einfach ohne Adresswahl, rechts: mit Festlegung der Adresse im I²C Datenbus.

LM75	Pin 1 SDA	Pin 2 SCL	Pin 3 OS	Pin 4 GND	Pin 5 A2	Pin 6 A1	Pin 7 A0	Pin 8 $+V_S$
J 8	Pin 3 GPIO 2	Pin 5 GPIO 3		Pin 9 GND	Pin 9 GND	Pin 9 GND	Pin 9 GND	Pin 1 3,3 V

Tabelle 1.15: Anschluss des MCP3208 an den SPI Datenbus

Werden mehrere Geräte am I²C Bus angeschlossen, müssen sie unterschiedliche Adressen haben. Über Pins 7, 6 und 5 des LM75 kann man seine Bus-Adresse im Wertebereich 0x48 (Pins 7, 6 und 5 alle auf low) bis 0x4F (Pins 7, 6 und 5 auf high) einstellen. Hierfür kann man einen dreipoligen DIP-Schalter und drei Pulldown-Widerstände (jeweils 10 kΩ), gemäß des Schaltplans in Abbildung 1.15 rechts, verwenden.

Die Aktivierung des I²C Busses wird ab Seite 375 erläutert. Pythonprogramme zur Temperaturmessung mit dem LM75 werden später vorgestellt, ab Seite 376.

Schaltung: Analogdatenerfassung mit MCP3208

Abbildung 1.22 zeigt eine Schaltung mit dem MCP320 (siehe Seite 31). Er wandelt eine analoge Eingangsspannung, die mit einem Potentiometer vorgegeben wird, in einen digitalen Datenstrom. Werden nur positive Spannungen eingelesen, liegen sie in einem Intervall von 0 V bis V_{REF}. In der Schaltung ist die Referenzspannung $V_{REF} = 3{,}3\,V$.

Die Anzahl der Bits eines AD-Wandlers gibt an, mit wie vielen Bits ein digitaler Datenwert gebildet wird. Bei n Bits sind 2^n verschiedene Datenwerte darstellbar. Der MCP3208 ist ein 12-Bit-AD-Wandler. Oft soll ein AD-Wandler andauernd Daten periodisch einlesen Die Frequenz, mit der das geschieht, heißt Abtastrate (sampling rate).

Analogspannungen beziehen sich auf die analoge Signalmasse, welche am Anschlusspunkt AGND vom Raspberry Pi zur Verfügung gestellt wird. Die Versorgungsspannung V_{DD} des MCP320 kann zwischen 2,7 V und 5,5 V liegen. Wir verwenden 3,3 V, die Spannung für die GPIO des Raspberry Pi. Der MCP3208 hat acht Analogeingänge (CH0 bis CH7). Wir verwenden nur CH2. Die Schaltung ist für sich genommen noch langweilig,

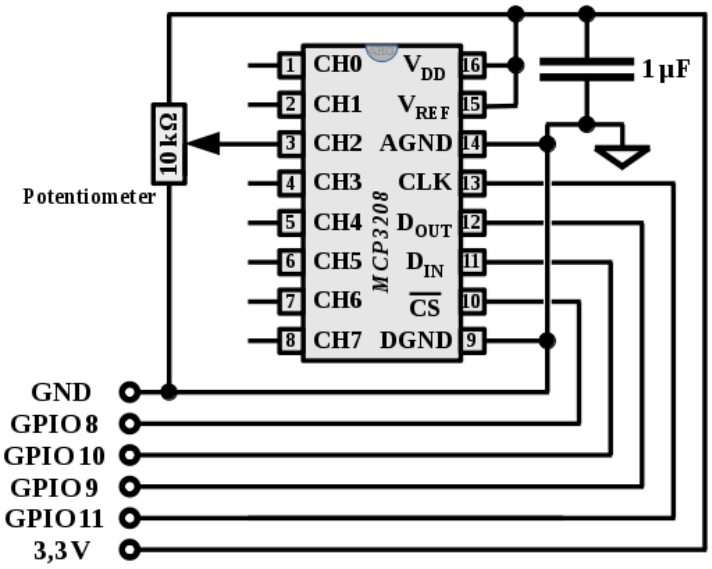


Abbildung 1.22: Analogdatenerfassung mit AD-Wandler MCP3208. Ein MLCC Vielschichtkeramikkondensator mit einer Kapazität von 1 µF wird nah am MCP3208 platziert, um die Versorgungsspannung zu stabilisieren. Tabelle 1.16 zeigt die Zuordnung der Anschlusspunkte zum SPI Datenbus.

aber sie kann als Grundlage für interessante Analogspannungsmessungen dienen.

Der MCP3208 tauscht digitale Daten mit dem Raspberry Pi über einen SPI Datenbus aus (siehe Seite 21). Dafür werden die mit CLK, $\overline{\text{CS}}$, D_{IN}, D_{OUT} bezeichneten Anschlusspunkte benutzt. DGND ist der Bezugspunkt (digitale Masse). In der Schaltung ist DGND mit AGND verbunden. Tabelle 1.16 zeigt die Zuordnung von Anschlusspunkten des MCP3208 zu SPI-Datenbus-Signalleitungen.

MCP3208 Anschlusspunkt	CLK	$\overline{\text{CS}}$	D_{IN}	D_{OUT}
SPI-Datenbus-Signalleitung	SCLK	$\overline{\text{CS}}$	MOSI	MISO

Tabelle 1.16: Anschluss des MCP3208 an den SPI Datenbus

Ein Pythonprogramm zur Spannungsmessung mit dieser Schaltung findet man im Abschnitt „A/D-Wandler MCP3208 am SPI Bus“ ab Seite 369.

2 Installation und Konfiguration

2.1 Grundkonfiguration

2.1.1 Installation, Konfiguration mit raspi-config

Das Betriebssystem des Raspberry Pi befindet sich traditionell auf einer Speicherkarte (SD-Karte oder microSD), die in seinem Speicherkarteneinschub steckt. Das ist einfach und platzsparend, aber nicht gut für einen dauerhaften Einsatz, weil diese Speicherkarten nicht so viele Schreibvorgänge aushalten wie eine Festplatte. Bei neueren Modellen ist es möglich, das Betriebssystem auf einem Speichermedium zu haben, das über USB angeschlossen ist, also auf einem USB-Stick oder einer Festplatte.

Für den Betrieb mit SSD Festplatte sind die Modelle 4 und 5 besonders geeignet. Sie haben bereits USB-Anschlüsse vom Typ 3 (in der Regel blau markiert). Über diese kann man eine SSD Festplatte mit Strom versorgen und man braucht meistens keinen USB Hub. Die Möglichkeit, das Betriebssystem über USB zu starten (booten), ist relativ neu. Es ist zu empfehlen, zunächst das Betriebssystem auf der Speicherkarte zu installieren und, wenn das Betriebssystem lauffähig ist, den ganzen Inhalt der Speicherkarte auf eine SSD zu kopieren (zu klonen), wie in Abschnitt 2.1.7 beschrieben (siehe Seite 64).

Nachdem das Image der Speicherkarte auf die SSD Festplatte geklont wurde, kann man den ganzen Speicherbereich der SSD Karte nutzbar machen, wie unten beschrieben wird (siehe Seite 45). Bei einem 32-bit Betriebssystem darf eine einzelne Datei höchstens knapp 4 GiB groß sein, aber die Festplatte kann viel größer sein.

Übertragung auf das Speichermedium

Die Raspberry Pi Foundation bietet auf der Webseite

`https://www.raspberrypi.com/software/operating-systems/`

das Betriebssystem (abgekürzt OS) *Raspberry Pi OS* auch in in der Version *Lite* an. Außerdem gibt es dort Hilfsprogramme, mit denen das OS in einfacher Weise auf die Speicherkarte (oder ein anderes Speichermedium) geschrieben werden kann.

Alternativ kann man es auf einem Rechner mit Linux-Betriebssystem auch „manuell“ (ohne besagte Hilfsprogramme) machen, indem

1. das OS als gepackte Image-Datei heruntergeladen wird
2. die gepackte Image-Datei entpackt wird
3. festgestellt wird, wie die Speicherkarte im Rechner heißt
4. Partitionen, die ihr OS vorschnell ins Dateisystem eingehängt („gemounted“) hat, ausgehängt („unmounted“) werden

5. die Image-Datei mit dem OS vom Rechner auf die Speicherkarte kopiert wird

Folgende Beschreibung setzt voraus, dass wir einen Linux-Rechner für diese Aufgaben verwenden. Beschreibungen für andere Betriebssysteme findet man im Internet.

1) Die gepackte Image-Datei vom 11. Dezember 2023 in der Betriebssystemversion Raspberry Pi OS Lite heißt `2023-12-11-raspios-bookworm-armhf-lite.img.xz` und benutzt das Datenkompressionsformat xz (Datei-Endung `.xz`). Wir laden die Datei herunter. Im folgenden nennen wir diese Datei kurz `OS.xz`. Im Verzeichnis von `OS.xz` kann man in einem Terminal mit

```
sha256sum OS.xz
```

eine Prüfsumme ausgeben. Bei fehlerfreier Übertragung sollte die Prüfsumme gleich der Angabe auf der Webseite der Raspberry Pi Foundation sein, die man erhält, wenn man auf `Show SHA256 file integrity hash:` klickt.

2) Zum Entpacken braucht man das Paket `xz-utils` auf dem Linux-Rechner. Durch

```
unxz OS.xz
```

wird entpackt und wir erhalten eine Datei mit der Endung `.img`. Fehlt das Paket `xz-utils` zum Entpacken, muss man es zunächst installieren.

3) *Bevor* wir die Speicherkarte einstecken, geben wir in einem Terminal den Befehl

```
lsblk -p
```

ein, um die Speichermedien des Rechners aufzulisten.

Wir stecken die Speicherkarte in das Lesegerät des Rechners und führen obigen Befehl erneut aus. Das zusätzlich angezeigte Gerät ist die Speicherkarte und könnte zum Beispiel `sda` heißen, muss aber nicht. Wir kürzen den Namen hier mit `X` ab und notieren ihn.

Falls die Speicherkarte partitioniert ist, werden darunter auch die Partitionen gezeigt und die die erste Partition könnte zum Beispiel `sda1` heißen, muss aber nicht. Wir kürzen Partitionsnamen hier mit `Y` ab.

4) In der rechten Spalte der Auflistung steht der „mountpoint“ für die eingehängten Partitionen. Partitionen der Speicherkarte sollten nicht eingehängt und die entsprechende Spalte leer sein. Falls Partitionen der Speicherkarte doch eingehängt sind, werden sie der Reihe nach mit dem Befehl `umount Y` ausgehängt, wobei `Y` der Name der Partition ist. Wir hängen aber nur die Partitionen der zu beschreibenden Speicherkarte aus!

5) Nun geben wir den Befehl zum Kopieren der Image-Datei mit Administratorrechten im Terminal ein. WICHTIG: Man muss darauf achten, dass als Ziel der Kopieraktion nur die Speicherkarte genannt wird, damit man nicht andere Speichermedien des Rechners überschreibt!

```
sudo dd bs=4M if=P of=/dev/X conv=fsync
```

Dabei ist `P` der Pfad zur Image-Datei und X der Name der Speicherkarte (nicht der Name einer Partition). Der Kopiervorgang dauert ein paar Minuten! Um sicherzustellen, dass der Kopiervorgang vollständig beendet wird, geben wir anschließend den Befehl

```
sync
```

ein. Danach ist die Übertragung des Betriebssystems auf eine Speicherkarte fertig.

Die Speicherkarte wird nun vorsichtig in den Einschub (*slot*) des Raspberry Pi geschoben, wobei die beschriftete Seite der Speicherkarte von der Platine abgewandt ist (nach unten zeigt) und die Seite mit den vergoldeten Kontakten der Platine zugewandt ist (nach oben zeigt). Die Speicherkarte muss richtig, das heißt: soweit ohne Gewalt möglich, eingesteckt sein. Bei einigen älteren Modellen des Raspberry Pin hat der Einschub einen federnden Auswurfmechanismus; bei neuen ist dieser nicht vorhanden.

Startvorbereitungen

Vor dem Einschalten des Raspberry Pi muss die Speicherkarte mit dem Betriebssystem eingesteckt sein (siehe oben, Seite 41), die Tastatur an einer beliebigen USB Buchse angeschlossen sein und der Bildschirm über die HDMI Buchse angeschlossen und eingeschaltet sein. Bei Raspberry Pi Modellen mit mehreren HDMI Buchsen nimmt man die mit `HDMI 0` beschriftete. Der Bildschirm sollte vor dem Raspberry Pi eingeschaltet sein, damit das Betriebssystem die Bildschirmauflösung automatisch bestens einstellen kann. Falls mehrere Gräte mit dem Bildschirm verbunden sind, kann es notwendig sein, im Menü des Bildschirms den Raspberry Pi als Quelle zu wählen. Alternativ kann man die anderen Geräte abtrennen. Ein LAN-Kabel soll zunächst *nicht* eingesteckt sein! Eine Verbindung mit einem Netzwerk und dem Internet wollen wir erst zulassen, wenn ein sicheres Passwort gewählt wurde.

Start und Konfiguration mit raspi-config

Der Raspberry Pi wird durch Anschluss der Versorgungsspannung eingeschaltet und das Betriebssystem wird automatisch geladen. Der Start des Betriebssystem heißt Booten (englisch booting). Dabei erscheinen Informationen über den Startvorgang.

Mit der Betriebssystemversion Raspberry Pi OS Lite vom 10. Oktober 2023 erscheint beim ersten Booten ein Fenster namens `Configuring keyboard-configuration`. Mit den Pfeiltasten der Tastatur bewegen wir die markierte (rot unterlegte) Zeile auf `Other` und drücken die ENTER Taste. Unter `Country of origin for the keyboard:` wählen wir in gleicher Weise `German` und unter `Keyboard layout:` schließlich nochmal `German`.

Danach folgt ein Fenster `Please enter new username:` , in dem wir den unseren Benutzernamen eintragen sollen. In früheren Versionen des Betriebssystems war der Benutzername `pi` vorgegeben, aber nun soll man selber einen Namen wählen. Ich wähle

```
ahg
```

Ich empfehle, einen Nutzernamen aus Kleinbuchstaben und ohne deutsche Sonderzeichen, wie Umlaute oder ß. Die Texteingabe wird mit der ENTER Taste abgeschlossen. Da gilt auch für die folgenden.

Im folgenden Fenster `Please set a password for ahg:` soll das Passwort für den Benutzer (hier: `ahg`) festgelegt werden. Auch hier sollte man zunächst nur Kleinbuchstaben ohne deutsche Sonderzeichen, wie Umlaute oder ß, verwenden. Wir wählen daher

vorläufig das einfache Passwort `qms` und setzen die Konfiguration damit fort. Es ist aber wichtig, dass man später ein neues, sicheres Passwort wählt, bevor man den Rechner mit einem Netzwerk und dem Internet verbindet! Wir tippen das vorläufige Passwort ein

```
qms
```

und drücken die ENTER Taste. Im folgenden Fenster `Please confirm the password:` wiederholen wir die Passworteingabe. Schließlich sollte die Zeile `raspberrypi login:` erscheinen und wir geben unseren Nutzernamen und das Passwort ein.

Danach sollte `raspberrypi:~ $` erscheinen. Diese Zeichenkette ist der Prompt. Er zeigt den angemeldeten Nutzer (`ahg`) und den Rechnernamen (raspberrypi). Eine blinkende Markierung am Zeilenende, sagt uns sagt, dass nach dem Prompt ein Befehl (als Zeichenkette) von uns erwartet wird. Wir befinden uns in der Konsole `tty1`.

Wir rufen nun das Programm `raspi-config` zur Konfiguration auf, um das Betriebssystem an unsere Bedürfnisse anzupassen. Der Befehl ist

```
sudo raspi-config
```

Der Vorsatz `sudo` ist nötig, da man das Programm nur mit Administratorrechten ausführen darf. Es erscheint eine textbasierte Nutzerschnittstelle mit einem Menü, in dem wir den Cursor (Eingabemarke) mit den Pfeiltasten ↑ und ↓ bewegen und einen Menüpunkt mit der ENTER Taste wählen können. Achtung: Das Programm reagiert langsam und manche Änderungen werden erst nach einem Neustart des Betriebssystems wirksam.

Wir nehmen zunächst die länderspezifischen Spracheinstellungen vor.

```
5 Localisation Options
```

Unter `L1 Locale` markieren wir durch Drücken der Leertaste das Auswahlfeld

```
de_DE.UTF-8 UTF-8
```

Im Feld vor der ausgewählten Option erscheint ein $*$. Das bereits mit einem $*$ versehene Auswahlfeld `en_GB.UTF-8 UTF-8` belassen wir, denn man kann mehrere dieser Spracheinstellungen haben. Durch Drücken der Tabulatortaste gelangt man auf das Feld `<OK>` und mit Drücken der ENTER Taste bestätigen wir die Auswahl. Nach `Default locale for the system environment:` wählen wir die Standardsprache aus den vorher ausgewählten länderspezifischen Spracheinstellungen. Für Englisch wählt man

```
en_GB.UTF-8
```

Wenn man lieber deutsch mit dem Computer spricht, kann man stattdessen `de_DE.UTF-8` wählen, sollte sich dann aber nicht über die „verdeutschte“ Computersprache ärgern. und bestätigen mit der ENTER Taste. Es dauert danach ein wenig.

Nochmal zu den länderspezifischen Einstellungen

```
5 Localisation Options
```

Unter `L2 Timezone` wählen wir

```
Europe
Berlin
```

Ein drittes Mal zu den länderspezifischen Einstellungen

```
5 Localisation Options
```

Unter `L3 Keyboard` wählen wir normalerweise

```
Generic 105-key PC  /  Gener. PC-Tastatur mit 105 Tasten
```

es sei denn, man hat eine besondere Tastatur. Unter `Keyboard layout:` kann man

```
German  /  Deutsch
The default for the keyboard layout  /  Der Standard für die Tastaturb.
Left Logo key
```

wählen. Beim letzten Punkt (`Compose key:`) bestimmt man eine Taste, mit deren Hilfe Sonderzeichen erzeugt werden, die nicht auf der Tastatur zu vertreten sind. Wenn man eine Taste hat, die ansonsten nutzlos ist, sollte man diese zur Compose-Taste machen. Auf vielen Tastaturen gibt es, links (`Left Logo key`) oder rechts (`Right Logo key`), eine Taste mit einer symbolischen Vierteilung, Windowstaste genannt.[1] Ich empfehle, `Left Logo key` zur Compose-Taste zu machen. Will man das nicht, wählt man stattdessen die Option `No compose key / Keine Compose-Taste`.

Allerdings wirkt auch die Tastenkombination `Ctrl .`, also gleichzeitiges Drücken der `Ctrl`-Taste (Strg) und der Taste für den Punkt (.), als Compose-Taste. Dafür muss nichts konfiguriert werden. Zur Anwendung der Compose-Taste siehe Seite 135.

Zur Vorbereitung von Wifi (WLAN) gehen wir ein viertes (und vorerst letztes Mal) zu den länderspezifischen Einstellungen

```
5 Localisation Options
```

Unter `L4 WLAN Country` wählen wir

```
DE Germany
```

Zur automatischen Anmeldung des Nutzers nach dem Systemstart wählen wir nun

```
1 System Options
S5 Boot / Auto Login
B2 Console Autologin
```

[1] Vierteilung ist eine mittelalterliche Foltermethode. Das Symbol ist ein Logo der Firma Microsoft.

Dann beenden wir `raspi-config` mit `<Finish>` (zwei Mal die Tabulatortaste und einmal die Entertaste drücken). Die Frage „Would you like to reboot now?“ beantworten wir mit `Yes`. Das System startet erneut. Ein Neustart heißt englisch reboot. Wenn wieder der Prompt erscheint, sollte die deutsche Tastatur richtig arbeiten. Falls der Neustart (reboot) hier nicht funktioniert, müssen wir den Rechner ausschalten (Trennung von der Spannungsversorgung) und wieder einschalten (Spannungsversorgung herstellen).

Nach dem reboot starten wir wieder das Konfigurationsprogramm.

```
sudo raspi-config
```

Statt den Befehl einzutippen, kann man ihn mit der Pfeiltaste ↑ aufrufen, denn er ist gespeichert, weil wir ihn bereits benutzt haben (siehe Abschnitt 4.1.1 auf Seite 110).

Wir geben dem Rechner einen Namen, unter dem er später im Netzwerk erkannt wird:

```
1 System Options
S4 Hostname
```

Der Rechnername darf Buchstaben des englischen Alphabets, Ziffern und einen Bindestrich enthalten. Ein Bindestrich darf aber nicht am Anfang oder Ende des Namens stehen. Da nach allgemeinen Regeln beim Rechnernamen, anders als sonst in Linux, nicht zwischen Groß- und Kleinschreibung unterscheiden werden soll, sollte man nur Kleinbuchstaben verwenden. Wir ändern `raspberrypi` in einen nicht zu langen, doch unterscheidbaren Namen, zum Beispiel `ahg-desk`

Nun wollen wir ein neues Passwort setzen, denn das alte (`qms`) kennt ja jeder.

```
1 System Options
S3 Password
```

und geben, wie verlangt, das neue Passwort zwei Mal ein. Wir beenden `raspi-config` über `<Finish>` und die Frage „Would you like to reboot now?“ bejahen wir.

Nach dem Start des Betriebssystems (Booten) starten wir das das Konfigurationsprogramm.

```
sudo raspi-config
```

WLAN kann man in einfachen Fällen mit `raspi-config` einrichten. Schwierigere Fälle werden auf Seite 53 erläutert. Für die einfache Einrichtung wählen wir unter

```
1 System Options
S1 Wireless LAN
```

und geben die Verbindungsdaten (SSID und Passwort) ein. Statt WLAN kann man auch, falls vorhanden, ein LAN-Kabel in die RJ-45 Buchse (siehe Abbildung 1.2) einstecken, um eine Verbindung zum Netzwerk und dem Internet herzustellen. Auf Seite 56 wird beschrieben, wie man die Verbindung zum Internet mit einem `ping` prüfen kann.

Als Audio-Ausgang kann man eine Monitorausgang oder die analoge Buchse wählen. Für den ersten HDMI- Ausgang wählen wir

```
1 System Options
S2 Audio
1 vc4-hdmi-0
```

Alternativ kann mit `...Headphones` die analoge Buchse gewählt werden.

Falls der Monitor einen schwarzen Rand zeigt, beseitigt man ihn mit

```
2 Display Options
D1 Underscan
```

Andernfalls ändert man dort nichts.

Wenn der Fernzugriff über SSH oder VNC erlaubt werden soll oder man einen Datenbus (SPI, I2C, Serial Port oder 1-Wire) verwenden will oder der Fernzugriff auf GPIO (siehe Seite 19) erlaubt werden soll, ermöglicht man die Benutzung unter

```
3 Interface Options
```

in den Punkten `I2 SSH` bis `I8 Remote GPIO`.

Beim ersten Booten sollte eine automatische Erweiterung des Dateisystems (automatic file system expansion) erfolgt sein. Dadurch kann die ganze Speicherkarte genutzt werden. Falls das nicht passiert ist oder man sicher sein will, dass es erfolgt ist. kann man folgendes wählen:

```
6 Advanced Options
A1 Expand Filesystem
```

Die Änderung wird aber erst nach dem nächsten Systemstart wirksam.

Nun sollte das System aktualisiert werden:

```
8 Update
```

Das erfordert eine Internetverbindung und dauert einige Zeit. Damit ist die Grundkonfiguration abgeschlossen. Erscheint der Prompt am Bildschirm, geben wir den Befehl

```
poweroff
```

um das Betriebssystem herunterzufahren. Danach schalten wir den Rechner durch Unterbrechung der Spannungsversorgung ganz aus. Will man den Rechner einschalten, legt man die Spannungsversorgung wieder an.

In früheren Versionen von `raspi-config` konnte man unter `4 Performance Options` in einem Menüpunkt `GPU Memory` die Größe des GPU Speichers einstellen. Dies fehlt in der aktuellen Version. Den aktuellen GPU Speicherwert erfährt man mit dem Befehl

```
vcgencmd get_mem gpu
```

Will man Videos abspielen, kann es vorteilhaft sein, der GPU mehr RAM zu geben, was aber den Anteil der CPU verringert, den man mit `vcgencmd get_mem arm` erfährt. Ohne Video-Anwendungen ist `64` für die GPU ausreichend. Sind Videos wichtig, ist `128` vermutlich besser. Der Wert für das GPU RAM kann in der Datei `/boot/config.txt` festgelegt werden. Dies erfordert Administratorrechte. Man öffnet dafür die Datei mit `sudo nano /boot/config.txt` und fügt am Ende zum Beispiel die Zeile `gpu_mem=76` an, gefolgt von einem Zeilenumbruch. Änderungen in der Datei `/boot/config.txt` werden erst nach einem Neustart (reboot) des Rechners wirksam. Man sollte jedoch sehr vorsichtig mit Änderungen in dieser Datei sein!

System neu starten oder beenden

Manchmal will man das Betriebssystem neu starten, zum Beispiel für die Übernahme von Änderungen. Dies geschieht mit dem Befehl

```
sudo reboot
```

Um anschließend den Bildschirm aufzuräumen, kann man den Befehl

```
clear
```

verwenden. Außer mehr Schwarz auf dem Bildschirm ändert sich dadurch aber nichts.

Um den Raspberry Pi auszuschalten, sollte man zuerst das System herunterfahren. Das Herunterfahren gibt dem System Zeit, Schreiboperationen auf den Datenträgern zu beenden. Es wird durch den Befehl

```
poweroff
```

bewirkt. Alternativ kann der Befehl `sudo shutdown -h now` benutzt werden. Erst danach, einige Sekunden später, nachdem die grüne LED im Rechner aufgehört hat zu blinken, unterbricht man die Stromversorgung. Wenn wir den Rechner wieder einschalten wollen, schließen wir die Stromversorgung wieder an.

Anpassung des display drivers in der Datei /boot/config.txt

Die Eigenschaften der GPU (beziehungsweise der Graphikkarte bei anderen Rechnern), welche den Bildschirm ansteuert, zum Beispiel Auflösung, Bildwiederholfrequenz und Farben) werden in neueren Linuxsystemen im Zuständigkeitsbereich des Kernels eingestellt. Dies nennt man *kernel modesetting (KMS)*.

Im Raspberry Pi OS wird dies ab der Version vom 30. Oktober 2021 eingesetzt. Leider bereitet der mit dem KMS eingeführte display driver `vc4-kms-v3d` oft Probleme bei der direkten Audioausgabe über ALSA, siehe Seite 7. Manchmal hilft die Installation des sound servers Pipewire (siehe Seite 78).

Ob eine Audioausgabe über HDMI möglich ist, kann man mit dem Konfigurationsprogramm `raspi-config` (siehe oben) prüfen. Unter `1 System Options` wählt man den

Punkt `S2 Audio` und erhält eine Liste der Geräte für die Audioausgabe (audio output). Wenn hier nur die eine Option `0 Headphones` steht, ist die Ausgabe über HDMI zunächst nicht möglich und es steht nur der analoge Audioausgang zur Verfügung.

Eine zuverlässigere Lösung ist die Bearbeitung der Datei `/boot/config.txt` mit einem Editor (zum Beispiel `nano`) und Administratorrechten, um einen anderen display driver zu verwenden, der erfahrungsgemäß gut mit ALSA zusammenarbeitet.

```
sudo nano /boot/config.txt
```

Die Verwendung des Texteditors `nano` wird in Abschnitt 4.2.1 auf Seite 131 beschrieben. Die Zeile `dtoverlay=vc4-kms-v3d` verwandelt man in

```
dtoverlay=vc4-fkms-v3d
```

mit einem `f` wie *firmware* (oder *fake*).

2.1.2 Größere Schrift, Unterverzeichnisse, Maus, copy & paste

Größere Schrift

In der Konsole kann man nicht die Schriften verwenden, die in einer graphischen Benutzeroberfläche verfügbar sind (GUI fonts), sondern muss sich mit einfacheren Konsolenschriften (console fonts) begnügen. Aber auch in der Konsole können Schriftart und -größe verändert werden, soweit die dafür benötigten *Fonts* verfügbar sind. Will man größere Zeichen, ruft man mit

```
sudo dpkg-reconfigure console-setup
```

ein Menü auf, in dem man zum Beispiel folgende Einstellungen vornimmt:

```
UTF-8
Guess optimal character set  /  Vermutlich optimaler Zeichensatz
Terminus
16 x 32 (nur Framebuffer)
```

Damit wir die Standardschrift für alle virtuellen Konsolen (virtual consoles tty1 – tty6) festgelegt. Man kann aber in jeder virtuellen Konsole die Schrift

Die zeichnerische Gestalt eines Schriftzeichens heißt Glyphe (siehe Seite 286). Mit

```
showconsolefont
```

werden alle Glyphen des Fonts gezeigt, der in der Konsole verwendet wird.

Jede virtuelle Konsole (virtual console, tty1 – tty6)

Unterverzeichnisse einrichten

Unser Home-Verzeichnis ist `/home/ahg`, wenn wir der Nutzer `ahg` sind. Eine Abkürzung für das Home-Verzeichnis des aktuellen Nutzers ist ~ (das Tilde-Zeichen). Um hier unsere Dateien gut geordnet ablegen zu können, richten wir mit dem Befehl `mkdir` Unterverzeichnisse ein, zum Beispiel `audio` für Audiodateien.

```
mkdir ~/audio
```

und entsprechend die Unterverzeichnisse `docum` (für Dokumente, zum Beispiel PDF-Dateien), `image` (für Bilddateien), `latex` (für LaTeX-Dokumente), `other` (für Alles, was nicht in ein anderes Verzeichnis passt), `progs` (für eigene Programme), `shell` (für Shellscripts), `texts` (für Textdateien) und `video` (für Videodateien). Es ist praktisch, dass die Namen dieser Verzeichnisse unterschiedliche Anfan gsbuchstaben haben, weil so die Namensergänzung mit Tab genutzt werden kann (siehe Seite 111).

Wir zeigen den Inhalt des Verzeichnisses mit

```
ls
```

an. Wenn das Paket `tree` installiert ist, (siehe Seite 222) kann man den Befehl `tree` verwenden, um die Verzeichnisstruktur übersichtlich darzustellen.

Maus mit copy & paste

In diesem Manuskript verwenden wir in der Regel keine Maus. Wenn man dennoch Eingaben mit einer Maus machen möchte, installiert man das Paket `gpm` mit

```
sudo apt install gpm
```

und nach dem nächsten Rechnerstart ist die Maus verfügbar. Nun muss dem Betriebssystem der Typ der Maus bekannt gegeben werden und auch weitere Einstellungen sind möglich (zum Beispiel für die Geschwindigkeit). Dazu kann man mit Administratorrechten in einem Texteditor die Konfigurationsdatei `/etc/gpm.conf` bearbeiten. Einfacher ist jedoch die Benutzung eines Konfigurationsprogramms. Es wird mit dem Befehl

```
sudo dpkg-reconfigure gpm
```

aufgerufen. Meistens kann man die Standardvorgaben des Konfigurationsprogramms durch Drücken der Enter-Taste übernehmen.

Als „Mouse device for GPM:“ kann `/dev/input/mice` bestätigt werden. Nach der Vorstellung der Maustypen muss man einen Typnamen nach „Mouse type:“ eingeben. Das kann zum Beispiel `ps2` sein. Leider gibt es viele Typen. Oft hat man mit alten Mäusen eher Erfolg. Nach der Konfiguration führt man einen Neustart des Rechners durch, um die Änderungen zu übernehmen.

Mit richtiger Konfiguration kann man eine Zeichenkette auf dem Bildschirm markieren, während man die linke Maustaste gedrückt hält. Ein Doppelklick markiert ein Wort. Die markierte Zeichenkette wird in einen Zwischenspeicher von `gpm` übertragen und

kann später durch Drücken der mittleren Maustaste irgendwo eingefügt werden (copy & paste). Das funktioniert in der Kommandozeile ebenso wie in einem Dokument, das mit einem Editor geöffnet wurde.

Bei manchen neueren Mäusen wir die markierte Zeichenkette erst durch gleichzeitiges Drücken der Tasten `Alt` und `c` in den Zwischenspeicher übertragen.

2.1.3 Umgebungsvariablen und Kurzbefehle setzen

Im Terminal, den wir nach Hochfahren des Betriebssystems ohne graphische Benutzeroberfläche sehen (Konsole), wird eine interaktive Login-Shell gestartet (bei uns die `bash`) und das Heimatverzeichnis des Nutzers (home directory) wird zum aktuellen Arbeitsverzeichnis (working directory). Der Pfad zum Heimatverzeichnis und zur Shell sind in der Datei `/etc/passwd` festgelegt. Für den Nutzer `ahg` sind das `/home/ahg` beziehungsweise `/bin/bash` (normalerweise).

Dient die bash als Shell, wird nach dem login ohne graphische Oberfläche (GUI) beim Starten der Shell (einmalig) die Datei `~/.bash_profile` ausgeführt, wenn sie existiert. Wenn, und nur wenn es `~/.bash_profile` nicht gibt, wird die Datei `~/.bash_login` ausgeführt, wenn sie existiert. Gibt es auch `~/.bash_login` nicht, wird `~/.profile` ausgeführt, wenn sie existiert, siehe Seite 108. Daraus folgt: Benutzt man die bash, kann man eine der drei genannten Dateien als Konfigurationsdatei verwenden. Wir nehmen die Datei `~/.profile`, weil sie (im Gegensatz zu `~/.bash_profile`) auch von manchen anderen Shells gelesen würde.

In einem Linux-Betriebssystem ohne graphische Benutzeroberfläche wird also beim Start der interaktiven Login-bash das Script ausgeführt, das in der versteckten Datei `.profile` steht. Das Script veranlasst unter anderem die Ausführung eines weiteren Scripts, das in der versteckten Datei `.bashrc` steht, siehe Seite 108. Beide Dateien liegen im Heimatverzeichnis des Nutzers, das allgemein mit ~ abgekürzt wird. Beim Nutzer `ahg` ist `/home/ahg/` der Pfad zum Heimatverzeichnis.

Beim login in einem Linux-Betriebssystem mit graphischer Benutzeroberfläche (GUI) wird die Datei `~/.profile` in der Regel durch den DisplayManager einmalig ausgeführt. Spätestens wenn eine Text-Shell (bash) verwendet wird, sollte bei allen Linux Betriebssystemen die Datei `~/.profile` ausgeführt werden. Die Datei `~/.bashrc` wird automatisch beim Start einer interaktiven Non-Login-bash gestartet. Dies geschieht in einem Linux-Betriebssystem mit graphischer Benutzeroberfläche beim Öffnen eines Terminalfensters, siehe Seite 109.

Zusammenfassung: Benutzt man die `bash` als Shell, kann ein Nutzer Einstellungen sowohl in der Datei `~/.profile` als auch in `~/.bashrc` vornehmen. Einstellungen, welche eine Subshell nicht automatisch erbt (`alias`-Befehle) und Funktionen), müssen in der Datei `~/.bashrc` vorgenommen werden (siehe Seite 50), nicht in der Datei `~/.profile`. Unveränderliche Umgebungsvariablen wie `PATH` sollten besser in `~/.profile` stehen. Umgebungsvariablen, die vielleicht von der `bash` geändert werden, sollten in `~/.bashrc` stehen.

.profile

Zur Bearbeitung der Datei `/.profile` mit Root-Rechten dient der Befehl

```
nano ~/.profile
```

Die Verwendung des Texteditors `nano` wird in Abschnitt 4.2.1 auf Seite 131 beschrieben. Die Datei `.bashrc` enthält Shell-Befehle.

Zur Ausführung eines Shellscripts (siehe Abschnitt 4.3 auf Seite 136) kann der Pfad zur Datei angegeben werden. Das ist jedoch umständlich.

Auf Seite 48 wird beschrieben, wie man ein Verzeichnis `shell` für Shellscripts einrichtet. Um Shellscripts, die in diesem Verzeichnis liegen, nur mit dem Dateinamen aufrufen zu können, teilen wir dem Betriebssystem mit, automatisch im eingerichteten Verzeichnis `shell` nach ausführbaren Dateien zu suchen.

Am Ende der geöffneten Datei fügen wir folgende Zeilen ein:

```
# set PATH so it includes user's directory shell if it exists
if [ -d "$HOME/shell" ] ; then
    PATH="$PATH:$HOME/shell"
fi
```

Pythonpakete, die mit dem Befehl `pip3 install -user ...` installiert werden, legen Dateien im Verzeichnis `/home/ahg/.local/bin` ab. Damit das Betriebssystem diese Dateien findet, muss die Umgebungsvariable `PATH` ergänzt werden. Die folgenden Zeilen sollten bereits in der Datei `/home/ahg/.profile` stehen:

```
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi
```

Nur wenn diese Zeilen noch nicht (an beliebiger Stelle) in der Datei `/home/ahg/.profile` enthalten sind, muss man sie am Ende einfügen.

In obigen Beispielen brauchen wir keinen `export`-Befehl, weil es `PATH` bereits als Umgebungsvariable gibt.

.bashrc

Zur Bearbeitung der Datei `~/.bashrc` mit Root-Rechten dient der Befehl

```
nano ~/.bashrc
```

Die Datei `.bashrc` enthält Shell-Befehle. Man kann hier vordefinierten oder eigenen Umgebungsvariablen Werte zuweisen. Wird einer Variablen, die noch keine Umgebungsvariable ist, ein Wert zugewiesen, muss sie exportiert werden (mit dem Befehl `export`), damit sie zu einer Umgebungsvariablen wird und damit für andere Programme (andere als die Shell) sichtbar ist. Die eigenen Ergänzungen sollte man am Ende der Datei anhängen und kann sie mit einem kurzen erklärenden Kommentar versehen. Der Kommentar muss mit einem Rautezeichen # beginnen und geht bis zum Zeilenende.

In einigen Anwendungsprogramme wird eine Datei mit einem Editor geöffnet. Mit den Umgebungsvariablen VISUAL und EDITOR können wir ein Standard-Editorprogramm festlegen, das verwendet wird, wenn nicht ausdrücklich ein anderer Editor angefordert wird. Dass es zwei Umgebungsvariablen zur Festlegung des Standard-Editors gibt, hat historische Gründe. Wenn `nano` unser Standard-Editor sein soll, schreiben wir

```
export EDITOR=nano
export VISUAL=nano
```

Ebenso legt man ein Standardprogramm zur seitenweisen Anzeige einer Textdatei auf dem Bildschirm mit der Umgebungsvariablen PAGER (Standard-Terminal-Pager) fest.

```
export PAGER=less
```

Als Standardprogramm für die Anzeige von PDF Dokumenten kann, nach Installation des Pakets `fbi`, das Programm `fbgs` dienen (siehe Seite 179).

```
export PDFVIEWER=fbgs
```

Als Standard-Browser legt man mit

```
export BROWSER=w3m
```

`w3m` (siehe Seite 209) fest. Alternativ ist natürlich auch `lynx` (siehe Seite 207) oder `links2` möglich (siehe Seite 210). Außerdem sollte man eine Startseite für den Browser `w3m` festlegen, also eine Webseite, die geöffnet wird, wenn `w3m` ohne Angabe einer Webseite aufgerufen wird. Die Suchmaschine DuckDuckGo wird durch

```
export WWW_HOME=https://duckduckgo.com/?kl=de-de
```

mit der Region Deutschland als Startseite festgelegt.

Im Wert der Umgebungsvariablen sollte keine Pfadangabe stehen, da einige Anwendungsprogramme nur Umgebungsvariablen ohne Pfadangabe richtig verarbeiten. Ist der Wert der Umgebungsvariablen eine Programmdatei, die ohne Pfadangabe nicht gefunden wird, kann man die Umgebungsvariable PATH erweitern, wie auf Seite 50 beschrieben.

Für einen langen, oft vorkommenden Befehl kann man einen Kurzbefehl mittels des Befehls `alias` definieren.

Muss man öfter die Partition `sda1` eines externen Datenträgers (zum Beispiel eines USB-Sticks) mounten (siehe Seite 90), kann folgende Definition helfen:

```
alias m='sudo mount -o uid=ahg,gid=ahg /dev/sda1 /media/usb_1
 && ls /media/usb_1/'
```

wobei `ahg` der Name des Nutzers ist. Obige Befehlszeile wurde hier aus Platzgründen in zwei Zeilen geschrieben; es soll aber nur eine Zeile sein! Die AND-Verknüpfung `&&` bewirkt, dass der zweite Befehl (`ls /media/usb_1/`) nur ausgeführt wird, wenn der erste (das Mounten) erfolgreich war. Auch für das Unmounten der Partition `sda1` kann ein Kurzbefehl definiert werden:

```
alias u='sudo umount /media/usb_1 && echo "Gerät unmounted"'
```

Wenn man das Anwendungsprogramm `ddgr` benutzt (siehe Seite 211), ist folgende Definition nützlich:

```
alias s='ddgr -n3 -rde-de -ty'
```

Ein gut gestalteter Bash-Prompt ist hilfreich. Er wird durch den Wert der Variablen `PS1` festgelegt. Im Betriebssystem *Raspberry Pi OS* wird PS1 in der Shell nur definiert, nicht exportiert. Daher ist PS1 eine lokale Variable und keine Umgebungsvariable, was aber praktisch unbedeutend ist, da PS1 nur in der Shell gebraucht wird. Es ist möglich und unschädlich, PS1 zu exportieren und damit zu einer Umgebungsvariablen zu machen.

Für die aktuelle Shell genügt eine Befehl wie

```
PS1=">"
```

um das Zeichen > als Bash-Prompt zu verwenden. Soll die Änderung des dauerhaft sein, also mit jedem neuen Terminal gesetzt werden, schreibt man die Befehlszeile ebenfalls ans Ende der Datei ∼`/.bashrc`.

Innerhalb der Zeichenkette, die den Bash-Prompt definiert, kann man unter anderem folgende Sonderzeichen und Variablen verwenden:

`\A`	Uhrzeit im Format `hh:mm` (24-Stunden), zum Beispiel `23:55`
`\d`	Datum `WW MMM DD` (Wochentag, Monat, Tag), zum Beispiel `Fr Apr 13`
`\H`	vollständiger Hostname, zum Beispiel `raspberrypi`
`\l`	Terminal auf dem die Shell läuft, zum Beispiel `tty1`
`\n`	Zeilenvorschub (neue Zeile, LF)
`\u`	Name des Nutzers (username), zum Beispiel `ahg`
`\w`	aktuelles Arbeitsverzeichnis , zum Beispiel ∼ (das entspricht `/home/ahg`)

Ein bunter Bash-Prompt ist möglich, aber auf die Farbgebung soll hier nicht eingegangen werden. Man kann auch Befehle der Shell benutzen.

```
PS1='$(pwd)' # Arbeitsverzeichnis, zum Beispiel /home/ahg, gefolgt von >
```

Wenn nur ein Nutzer das Betriebssystem verwendet, ist eine Angabe des Nutzer im Bash-Prompt nicht sinnvoll. Ebenso braucht man nicht jedes Mal den Hostnamen. Außerdem sollte der Bash-Prompt nicht zu lang sein. Ein Vorschlag:

```
PS1='\n\A \l $(pwd) > ' # Zeilenvorschub, Zeit, Konsole, Arbeitsverz. >
```

Man beachte, dass das `l` in `\l` der Kleinbuchstabe l und nicht die Ziffer 1 ist.

2.1.4 WLAN Konfiguration, Fernsteuerung über SSH

Die Einrichtung einer WLAN Verbindung erfolgt in der Regel über `raspi-config` und den NetworkManager, der bereits installiert ist (Paket network-manager).

WLAN Einrichtung über raspi-config und den NetworkManager

Die grundlegende Einrichtung über `raspi-config` wurde auf Seite 44 beschrieben. Das Programm `nmcli` ist die Schnittstelle in der Kommandozeile zur Steuerung und Kontrolle des NetworkManager. Eine schöne Anleitung findet man unter

`https://wiki.ubuntuusers.de/NetworkManager/NetworkManager_ohne_GUI/`

Die verfügbaren WLAN-Netzwerke (wifi) werden mit dem Befehl

```
nmcli dev wifi list
```

angezeigt, wobei ein ∗ die aktive Verbindung markiert. Will man nur die SSID des aktiven WLAN Netzwerks auf dem Bildschirm ausgeben, gelingt dies mit

```
nmcli connection show --active | grep wifi | awk '{print $1}'
```

Hat man das Passwort für das aktive WLAN Netzwerk vergessen, kann man es sich mit

```
nmcli device wifi show-password | grep Password | awk '{print $2}'
```

anzeigen lassen.

Alternative WLAN Einrichtung

Wenn, aus besonderen Gründen, die automatische WLAN Einrichtung über den NetworkManager nicht möglich ist, kann man die Konfigurationsdatei für WLAN-Netze direkt anpassen. Damit ist man flexibler und kann zum Beispiel auch Verbindungen über *eduroam* einrichten. Statt einer kann man auch mehrere WLAN-Verbindungen einrichten. Sind mehrere WLAN-Netze erreichbar, kann die Auswahl des WLAN-Netzes automatisch über das Betriebssystem erfolgen oder man kann dies auch selbst steuern.

Erreichbare WLAN Netzwerke kann man mit dem Befehl

```
iwlist wlan0 scan | less
```

auflisten lassen.

Zur WLAN Konfiguration werden die Dateien und `/etc/network/interfaces` und `/etc/wpa_supplicant/wpa_supplicant.conf` mit einem Texteditor wie *nano* (siehe Seite 131) verändert. Dies erfordert Administratorrechte.

Wir beginnen mit der Datei `interfaces` . Hier werden die Regeln für den sogenannten ifupdown-Mechanismus festgelegt. Der *NetworkManager* für Systeme mit graphischer Benutzeroberfläche wird *nicht* verwendet.

```
sudo nano /etc/network/interfaces
```

Die Datei soll folgenden Inhalt haben:

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

# Ergaenzung
auto lo
iface lo inet loopback
iface eth0 inet dhcp
auto eth0
allow-hotplug wlan0
auto wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Die Zeilen `auto eth0` und `auto wlan0` bewirken, dass die Schnittstellen `eth0` und `wlan0` beim Start des Rechners durch den Init-Prozess aktiviert werden. Braucht man eine der Schnittstellen nicht, kann man die entsprechende Zeile weglassen und die betreffende Schnittstelle später bei Bedarf „manuell“ aktivieren (mit dem `ifup`-Befehl).

Nun verändern wir die Datei `wpa_supplicant.conf` .

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Am Anfang der Datei müssen folgende Zeilen stehen:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=DE
```

Eine neue WLAN Verbindung wird eingerichtet, indem am Ende der Datei einige Befehlszeilen angefügt werden. Nach der Änderung verlässt man *nano* mit der Tastenkombination Strg-X und bestätigt die Speicherung mit Y für Yes. Die Änderung wird wirksam, nachdem das Betriebssystem neu gestartet wurde. Die hier gezeigte Einrichtung von WLAN-Netzen speichert das Passwort unverschlüsselt. Wer Zugang zum Rechner hat, kann die WLAN-Passwörter also auslesen. Will man eine sichere Einrichtung, muss man andere Anleitungen verwenden. Wir beschreiben hier nur die einfache Lösung.

WLAN mit SSID (nicht versteckt) und Passwort In den meisten Fällen will man eine WLAN-Verbindung einrichten, die mit WPA und Passwort verschlüsselt ist. Man braucht den Namen des WLAN-Netzwerks (SSID) und das Passwort. Wenn das Netzwerk nicht versteckt ist, also der Name bei der Suche nach Netzwerken erscheint, hängt man folgende Zeilen an das Ende der Datei `/etc/wpa_supplicant/wpa_supplicant.conf` .

```
network={
    ssid="WLAN-Netzwerkname"
    psk="WLAN-Passwort"
    id_str="mein_Netzwerkname"
}
```

Dabei ist *mein_Netzwerkname* ein beliebiger Name, der die Netzwerke in Listen unterscheidbar macht. Jedes Netzwerk bekommt einen eigenen Wert für *mein_Netzwerkname*!

WLAN mit SSID (versteckt) und Passwort Ist der Name des Netzwerks versteckt, hängt man Folgendes an das Ende von `/etc/wpa_supplicant/wpa_supplicant.conf` .

```
network={
    ssid="WLAN-Netzwerkname"
    scan_ssid=1
    psk="WLAN-Passwort"
    id_str="mein_Netzwerkname"
}
```

Dabei ist *mein_Netzwerkname* wieder ein einzigartiger, ansonsten beliebiger Name.

eduroam Für eine Verbindung über *eduroam* genügt es manchmal, Folgendes an das Ende der Datei `/etc/wpa_supplicant/wpa_supplicant.conf` anzufügen.

```
network={
        ssid="eduroam"
        proto=RSN
        key_mgmt=WPA-EAP
        eap=PEAP
        identity="eduroam-Name"
        password="eduroam-Passwort"
        phase1="peaplabel=0"
        phase2="auth=MSCHAPV2"
        id_str="EduRoam"
}
```

Dabei sind *eduroam-Name* und *eduroam-Passwort* die persönlichen Zugangsdaten und die nach `id_str` angegebene Zeichenkette ein einzigartiger, ansonsten beliebiger Name.

wicd-curses

Zur übersichtlichen Darstellung empfangbarer Funknetzwerke und zur Einstellung einer WLAN Verbindung kann man auch das Programm `wicd-curses` benutzen, das mit

```
sudo apt install wicd-curses
```

installiert wird. Allerdings ist dies in den meisten Fällen überflüssig.

`wicd-curses` erzeugt eine Tabelle, in der die Empfangsstärke, Verschlüsselungsart und Kanalnummer der Funknetzwerke aufgelistet werden. Eingabe von `R` (groß geschrieben) über die Tastatur erneuert die Anzeige.

Prüfung der Verbindung zum Internet

Um zu prüfen, ob die WLAN Verbindung zum Internet funktioniert, kann man mit

```
ping -c 1 -W 1 195.191.15.153
```

eine (belanglose) Datenanfrage an die Webseite von Google senden. Die Option `-c 1` bestimmt, dass nur 1 Paket (ECHO_REQUEST) gesendet wird und `-W 1` bestimmt, dass nur 1 Sekunde auf eine Antwort (ECHO_REPLY) gewartet wird.

Starten des SSH Dienstes beim Raspberry Pi

Zum Start von SSH während der aktuellen Sitzung des Raspberry Pi wird der Befehl

```
sudo systemctl start ssh
```

eingegeben und zum Abschalten während der aktuellen Sitzung

```
sudo systemctl stop ssh
```

Damit SSH zukünftig automatisch gestartet wird, gibt man

```
sudo systemctl enable ssh
```

ein und mit dem Befehl

```
sudo systemctl disable ssh
```

wird SSH zukünftig nicht mehr automatisch gestartet. Den Status erfährt man mit

```
sudo systemctl status ssh
```

Um den Raspberry Pi über das Netzwerk ansprechen zu können, benötigen wir seine IP-Adresse. Diese erfahren wir mit dem Befehl

```
hostname -I | awk '{print $1}'
```

Das erste Wort der Ausgabe von `hostname -I` ist die IP-Adresse. Durch das Zeichen `|` wird die Ausgabe von `hostname -I` an das Programm `awk` weitergeleitet, welches sie in einzelne Worte zerlegt. `$1` nimmt das erste Wort, das mittels `print` ausgegeben wird.

Um unseren Raspberry Pi über das Netzwerk von einem anderen Rechner mit Linux zu bedienen, benutzen wir den Befehl `ssh` im Terminal des anderen Rechners.

```
ssh ahg@192.168.2.173
```

Dabei ist `ahg` der Name eines Nutzers des Raspberry Pi und `192.168.2.173` ist die IP-Adresse, wobei hier natürlich die aktuellen Werte einzusetzen sind. Im anschließenden Dialog gibt man das Passwort des Nutzers ein und wird dann verbunden.

Der gleiche Nutzer (hier: `ahg`) kann gleichzeitig über mehrere Terminals angemeldet sein, zum Beispiel am Raspberry Pi über die virtuelle Konsole `tty1` und am anderen Rechner über das Pseudoterminal `pts/0`.

Zum Beenden der SSH-Verbindung schließt man das Terminal des anderen Rechners mit dem Befehl `exit`.

2.1.5 Papiergröße, Scanner und Drucker, Zeiteinstellung

Papiergröße

Um Dienst- und Büroprogrammen mitzuteilen, dass unser bevorzugtes Papierformat DIN A4 ist, installieren wir das Paket `libpaper-utils` mit dem Befehl

```
sudo apt install libpaper-utils
```

und geben danach den Befehl

```
sudo paperconfig -p a4
```

Scanner anschließen

Einfach ist die Verwendung eines separaten Scanners (kein Multifunktionsgerät), der über USB angeschlossen wird und eine Schnittstelle zu SANE (Scanner Access Now Easy) hat. Folgendes wurde mit dem CanoScan LiDE 300 erfolgreich getestet.

Zunächst installiert man SANE mit

```
sudo apt install libsane
```

und schließt den Scanner an. Eventuell muss man eine Sperre für den beweglichen Schlitten des Scanners öffnen, bevor man den Scanner einschaltet. Wenn der Scanner kein Netzteil hat, sondern elektrische Leistung über den USB-Anschluss bezieht, sollte man einen aktiven USB-Hub nehmen oder zumindest eine USB 3 Buchse, die mehr Leistung abgeben kann als eine USB 2 Buchse. Dann prüft man, ob der Scanner erkannt wird

```
scanimage -L
```

und kann danach scannen. Die Knöpfe auf dem Gerät braucht man nicht. Der Befehl

```
scanimage >image.pnm
```

sollte eine Datei namens `image.pnm` erzeugen, die den Scan als Bild im PNM-Format enthält. Man kann die Datei mit einem Bildbetrachtungsprogramm wie `fbi` (siehe Seite 173) oder `fim` (siehe Seite 173) ansehen. Mit dem Programm `pnmtopng` kann man aus der PNM-Datei eine neue Datei im PNG-Format machen:

```
pnmtopng image.pnm > image.png
```

Man kann die Steuerung des Scanners über SANE auch mit einem Pythonprogramm durchführen, siehe Seite 338, und zum Beispiel die erzeugte Datei automatisch anzeigen.

Drucker anschließen

CUPS (Common Unix Printing System) ist das am häufigsten verwendete Programmpaket für die Steuerung von Druckern unter Linux. Eine deutsche Anleitung erhält man mithilfe eines Browsers, zum Beispiel `w3m` (siehe Seite 209) durch den Befehl

```
w3m /usr/share/doc/cups/online-docs/de/index.html
```

Mit `q` kann man den Browser `w3m` verlassen.

Jeder Drucker braucht einen Treiber. Dieser besteht aus einer Textdatei mit der Endung `.ppd`, welche Eigenschaften und Fähigkeiten des Druckers beschreibt, und einem sogenannten Filterprogramm, welches die Daten für den Drucker aufbereitet. Ergänzend können Dateien für Farbdruck und Hilfsfunktionen hinzukommen. CUPS enthält Filterprogramme für viele Drucker, benötigt aber meistens eine `.ppd`-Datei. Für Drucker, die in einem Netzwerk über AirPrint angesprochen werden, stellt CUPS jedoch den Treiber `IPP Everywhere` bereit, der keine `.ppd`-Datei braucht.

Zunächst installieren wir CUPS, und bei Bedarf auch Pakete für Druckertreiber

```
sudo apt install cups cups-client printer-driver-gutenprint hplip
```

Wenn man keinen HP-Drucker hat, lässt man `hplip` in obigem Befehl weg. Hat man für einen HP-Drucker bereits eine `ppd`-Datei für den Druckertreiber, ist es meistens besser, diese zu benutzen und auf `hplip` zu verzichten.

Nun fügen wir den Nutzer ahg (das sind wir) zur Gruppe `lpadmin` hinzu

```
sudo usermod -a -G lpadmin
```

damit man ohne Administratorrechte (also ohne `sudo`) Druckereinstellungen ändern darf (G groß geschrieben; `-a -G` steht für add group). Nun wird das System neu gestartet.

```
sudo reboot
```

Nach der Installation von CUPS und der Gruppenänderung sollte man das Betriebssystem neu starten, damit CUPS zur Verfügung steht. CUPS ist ein Client/Sever-System das automatisch gestartet wird. Den Zustand erfährt man mit dem Befehl

```
sudo systemctl status cups
```

Die Konfiguration von CUPS kann über Befehle im Terminal und eine lokale (im Rechner gespeicherte) Webseite (web interface) von CUPS erfolgen. Das web interface von CUPS erreicht man unter der URL `http://localhost:631`. Bei Betriebssystemen mit graphischer Oberfläche gibt es oft besondere Konfigurationsprogramme, aber die brauchen wir nicht.

Die Konfiguration eines Druckers ist nicht immer einfach, was sowohl am Drucksystem CUPS, als auch an den Druckerherstellern liegt, welche nicht gern Treiberprogramme für Linux bereitstellen. Wir beschränken uns auf einen HP-Drucker, den wir lokal über USB mit dem Raspberry Pi verbinden.

Wir verwenden den Browser `link2` im Graphikmodus (siehe Seite 210). Alternativ geht es auch mit `lynx` (siehe Hinweis auf Seite 60). Wir verbinden wir den Drucker über USB mit dem Raspberry Pi und schalten ihn ein. Falls sich der Drucker im weiteren Verlauf selbst ausschaltet (um Energie zu sparen), muss man ihn natürlich wieder einschalten.

Vorteilhaft ist, wenn das Paket `printer-driver-gutenprint` einen Treiber für den Drucker bereitstellen kann. Für HP-Drucker gibt es das Paket `hplip`.

In `hplip` sind Konfigurationsprogramme und Treiberdateien enthalten. Eine Internetverbindung ist oft erforderlich, da die Programme Druckertreiber und zusätzliche Firmware aus dem Internet herunterladen, wenn diese nicht in `hplip` enthalten sind.

Die Erfahrungen damit sind sehr unterschiedlich. Die Konfiguration eines HP-Druckers kann über das Programm `hp-setup` durchgeführt werden (getestet: `HP Laserjet 1200`, welcher nur eine `.ppd`-Datei benötigt), welches mit der Option `-i` in der Konsole ausgeführt werden kann. Das Programm sollte mit Administratorrechten gestartet werden:

```
sudo hp-setup -i
```

Es gibt Drucker, die neben der `ppd`-Treiberdatei auch zusätzliche Firmware (Plugins) benötigen. Falls `hp-setup` diese nicht selbst herunterlädt und installiert, kann das Programm `hp-plugin`, falls vorhanden, verwendet werden. Dieses wird auch mit der Option `-i` in der Konsole, aber ohne Administratorrechte ausgeführt:

```
hp-plugin -i
```

Das Programm lädt die zusätzlichen Plugins herunter. Wenn der Download nicht erfolgreich ist, können die Dateien auch manuell von HP heruntergeladen werden:
https://developers.hp.com/hp-linux-imaging-and-printing/plugins
Sowohl die `.run`-, als auch die `.run.asc`-Datei sind dann notwendig (erfolgreich getested mit dem Druckertyp `HP Laserjet P1109w`).

Treiber mit der Endung `.ppd` werden automatisch im Verzeichnis `/etc/cups/ppd/` gespeichert. Man sollte das Verzeichnis kopieren und dauerhaft auf einer Festplatte speichern, damit man bei späteren Installationen die ppd-Dateien verwenden kann.

Nun starten wir das web interface von CUPS

```
links2 -g http://localhost:631
```

und ärgern uns nicht über die Macken. Im Abschnitt `CUPS für Administratoren` aktivieren wir mit Pfeiltasten und ENTER die Option

```
Drucker und Klassen hinzufügen
```

Falls dann statt einer „gerenderten“ Webseite ein HTML-Quelltext angezeigt wird, ärgern wir uns nicht (siehe oben), sondern drücken die Backslash-Taste

```
\
```

Im Abschnitt `Drucker` aktivieren wie `Drucker hinzufügen`. CUPS fragt uns nach Benutzernamen (`pi`) und Passwort. Danach können wir unseren lokalen Drucker auswählen, wobei man die Klasse HP und nicht USB nimmt. Entsprechend kann man nun die weiteren Einstellungen vornehmen. Nicht vergessen sollte man, den neuen Drucker zum Standarddrucker (default printer) zu machen, denn ansonsten müsste man bei jedem Druckauftrag den Drucker angeben und dessen Name ist lang.

Wenn man alles eingestellt und den Browser geschlossen hat, kann man eine Textdatei drucken. Wenn die Datei `f.dat` heißt, genügt

```
lp f.dat
```

und es sollte gedruckt werden. Mit `-o media=a4` legt man die Seitengröße A4 fest, wenn das nicht schon der Standardwert für den Drucker ist. In den USA übliche Alternativen zu `a4` sind `letter` oder `legal`. Mit `-P` können die zu druckenden Seiten festgelegt werden, falls nicht alles gedruckt werden soll, zum Beispiel Mit `-P 1-3,6` für die Seiten 1, 2, 3 und 6.

Wenn man den Browser `lynx` (siehe Seite 207) zur Administration von CUPS verwendet, kann es auch vorkommen, dass Seiten nicht „gerendertert“ erscheinen, sondern der Quelltext angezeigt wird. Wieder hilft dann die Backslash-Taste \ (zweimal).

Statt mit dem web interface kann man CUPS auch mit Befehlen steuern. Durch

```
sudo lpinfo -v | grep "direct usb"
```

erhält man zum Beispiel eine Liste der über USB angeschlossenen Drucker, in der jede Zeile mit `direct usb` beginnt. Beim Drucken mit CUPS werden die zu druckenden Daten verarbeitet und an einen geeigneten (zum Drucker passenden) CUPS-Ausgang, *backend* genannt, gesandt. Das backend sendet die Daten weiter zum Drucker. Das backend `direct usb` ist für Drucker zuständig, die direkt über USB angeschlossen sind.

Zeiteinstellung

Das aktuelle Datum und die Uhrzeit, wie sie im Betriebssystem vorliegen (Systemzeit), erfährt man mit dem Befehl

```
date
```

Der Raspberry Pi hat eine innere Uhr, die nur läuft, wenn er eingeschaltet ist. Wird er ausgeschaltet, verliert er das richtige Datum und die richtige Uhrzeit. Besteht eine Internet-Verbindung, holt der `systemd-timesyncd.service` nach dem Einschalten die aktuelle Zeit von einem *time server*. Mit dem Befehl

```
timedatectl
```

erfährt man den Status (Zustand) vom `systemd-timesyncd.service`.

Ohne Internetverbindung setzt man Datum und Zeit selbst. Dafür kann man den `systemd-timesyncd.service` oder den Befehl `date` verwenden.

Mit dem `systemd-timesyncd.service` inaktiviert man zunächst die automatische Zeiteinstellung über das Internet durch

```
sudo timedatectl set-ntp false
```

und setzt dann Datum und Zeit, zum Beispiel auf den 25. Oktober 2021, 14 Uhr 33.

```
sudo timedatectl set-time '2021-10-25 14:33:00'
```

Falls man die automatische Zeiteinstellung wieder aktivieren will, gibt man

```
sudo timedatectl set-ntp true
```

ein.

Alternativ kann man mit dem Befehl

```
sudo date -s '2021-10-25 14:33:00'
```

Datum und Zeit setzen, im Beispiel auf den 25. Oktober 2021, 14 Uhr 33.

Obwohl die automatische Zeiteinstellung mit Internetverbindung meistens problemlos funktioniert, kann sie versagen, wenn der Raspberry Pi in einem besonderen Netzwerk betrieben wird. Erfahrungsgemäß kann es helfen, wenn man (mithilfe eines Texteditors wie `nano`) die versteckte Datei `/home/ahg/.profile` am Ende wie folgt ergänzt.

```
sudo date -s "$(wget -qSO- --max-redirect=0 www.google.com 2>&1 | \
grep Date: | cut -d' ' -f5-8)Z"
```

2.1.6 Konfigurations-, Informations- und Gerätedateien

Die folgende Liste beschreibt kurz verschiedene Dateien, die zur Konfiguration des Betriebssystems verwendet werden oder Daten bereithalten, welche der Information des Nutzers dienen können. Eine kommentierte Liste der Unterverzeichnisse des Hauptverzeichnisses / steht weiter unten (siehe Seite 91).

Mögliche Änderungen der Dateien werden hier nicht erörtert, aber es gibt einen allgemeinen Hinweis hierzu: In Konfigurationsdateien sollte auch die letzte nicht-leere Zeile mit einem Zeilenvorschub abgeschlossen werden, da sie sonst nicht von allen Programmen richtig gelesen wird. Außerdem werfen wir einen Blick auf Gerätedateien.

Verzeichnis /boot

/boot/cmdline.txt enthält eine Zeile mit Angaben, die der Kernel beim Booten liest. Unter anderem wird zum Beispiel mit `root=/dev/mmcblk0p2` angegeben, dass sich die Wurzel des Dateisystems `/` in Partition 2 der Speicherkarte `mmcblk0` befindet, falls von einer SD-Karte oder microSD dieses Namens gebootet wird.

Will man die Bildschirmanzeige nach einer bestimmten Zeit ohne Aktivität, zum Beispiel nach 600 Sekunden (10 Minuten), über das Betriebssystem automatisch ausschalten, kann man, nach einem Leerzeichen (ohne Einfügen eines Zeilenumbruchs), die Zeichenkette `consoleblank=600` anhängen. Entsprechend wird mit `consoleblank=0` die Abschaltung der Bildschirmanzeige über das Betriebssystem verhindert.

In anderen Linux-Systemen gibt es oft eine entsprechende Datei `/proc/cmdline`.

/boot/config.txt legt Werte für die Konfiguration der Hardware fest. Die Datei ersetzt beim Raspberry Pi teilweise die bei anderen Rechnern übliche Firmware UEFI (oder früher BIOS), welche es beim Raspberry nicht gibt. Beim Start des Rechners liest die GPU die Datei config.txt, bevor die CPU initialisiert wird und bevor Linux geladen wird. Einige Werte, die in der config.txt eingestellt sind, können mit dem „Werkzeug“ `vcgencmd` und folgenden beiden Befehlen gelesen werden:

```
vcgencmd get_config int | less
vcgencmd get_config str
```

Allerdings kann `vcgencmd` noch mehr als nur Information über config.txt zu liefern.

Verzeichnis /dev

Periphere Geräte (englisch peripheral devices) sind körperliche (nicht virtuelle) Eingabegeräte (input devices), die Daten zur CPU senden, oder Ausgabegeräte (output devices), die Daten von der CPU empfangen.

Der Datenaustausch mit peripheren oder virtuellen (siehe unten) Geräten wird durch besondere Programme gesteuert, die *Gerätetreiber* (device driver), kurz Treiber (driver), genannt werden. Der Zugang zu einem Gerätetreiber wird vom Linux Kernel durch *Gerätedateien* (device files) ermöglicht. Das sind dateiähnliche Objekte (file-like objects), die sich aus Sicht eines Nutzerprogramms ähnlich wie echte Dateien verhalten. Das Nutzerprogramm kann aus der Gerätedatei eines Eingabegeräts lesen (read) und in die Gerätedatei eines Ausgabegeräts schreiben (write).

Der Kernel von Linux stellt außerdem Treiber und Gerätedateien für virtuelle (nichtkörperliche) „Geräte“, sogenannte *Pseudo-Geräte* (virtual devices) zur Verfügung. Sie stellen nicht ein peripheres Gerät dar, sondern verhalten sich nur so. Alle Gerätedateien (für periphere Geräte und Pseudo-Geräte) stehen in der Regel im Verzeichnis `/dev`.

Hinsichtlich der Dateneingabe oder Datenausgabe unterscheidet man zwei Typen von Gerätedateien. Sogenannte zeichenorientierte Gerätedateien (englisch character devices) arbeiten ohne Puffer (buffer), Typabkürzung `c`, und blockorientierte Gerätedateien (block devices) mit Pufferung, Typabkürzung `b`.

/dev/null ist eine virtuelle, leere, zeichenorientierte Gerätedatei, in die man schreiben kann (write). Alles wird verworfen, aber formal ist der Schreibvorgang erfolgreich. Unerwünschte Datenströme werden so umweltschonend entsorgt.

/dev/random ist eine virtuelle, zeichenorientierte Gerätedatei, aus der man beliebig viel lesen kann (read). Geliefert werden zufällige Bytes. Die meistens bessere Alternative ist `/dev/urandom` (siehe unten).

/dev/tty ist eine zeichenorientierte Gerätedatei, in die man lesen (read) und schreiben kann (write). Sie repräsentiert das Terminal, in dem der aktuelle Prozess abläuft.

Ein Beispiel zur Veranschaulichung: Der Befehl `echo "Hallo" > /dev/tty` schreibt `Hallo` auf den Bildschirm.

/dev/urandom ist eine virtuelle, zeichenorientierte Gerätedatei, aus der man beliebig viel lesen kann (read). Geliefert werden zufällige Bytes vom Zufallszahlengenerator des Kernels. In den meisten Fällen ist `/dev/urandom` besser geeignet als `/dev/random` (siehe oben), weil sie nicht blockiert.

/dev/zero ist eine virtuelle, zeichenorientierte Gerätedatei, aus der man beliebig viel lesen kann (read). Geliefert werden Nullzeichen (englisch null character), abgekürzt NUL (nicht die Ziffer 0). Sie haben keine Glyphe (siehe Seite 286) und sind auf dem Bildschirm nicht zu sehen.

Ein Beispiel zur Veranschaulichung: Der Befehl `tr '\000' '\101' < /dev/zero` erzeugt einen Datenstrom aus Nullzeichen, `'\000'`, in dem jedes Nullzeichen durch den Großbuchstaben A, `'\101'`, ersetzt wird. Auf der Standardausgabe (Bildschirm) wird eine nicht endende Folge von A ausgegeben, bis der Spuk durch `Ctrl-C`, das heißt: gleichzeitiges Drücken der Tasten `Ctrl` (oder `Strg`) und `c`, beendet wird (siehe Seite 12).

Verzeichnis /etc

/etc/default/keyboard enthält Angaben zur Tastatur, insbesondere Typ (Modell) und Sprachbezug (Layout), zum Beispiel `de` für eine Tastatur mit deutschen Umlauten. Eine Änderung sollte nicht mit einem Editor geschehen, sondern mit dem Befehl

```
sudo dpkg-reconfigure keyboard-configuration
```

oder über `raspi-config` (siehe Seite 42).

/etc/machine-ID enthält eine Zeichenkette (machine ID), die während der Installation oder dem ersten Start des Betriebssystems als zufällige Zeichenfolge erzeugt wurde. Aufgrund der Länge der Zeichenkette ist die Wahrscheinlichkeit, einen zweiten Rechner mit gleicher machine ID zu finden, verschwindend gering. Die machine ID sollte niemals nach außen (über das Netzwerk) bekannt gegeben werden. Einige Programme verwenden die machine ID, um den Rechner (genauer: dessen aktuelles Betriebssystem) über das Netzwerk in verschlüsselter Form zu identifizieren. Dabei bleibt die machine ID geheim.

/etc/mplayer enthält Konfigurationsdateien für den Mediaplayer MPlayer, welche für alle Nutzer gelten (`input.conf`, `menu.conf`, `mplayer.conf`). Der Nutzer `pi` kann zusätzlich eine Datei ∼`/.mplayer/config` haben.

/etc/os-release enthält Namen und Version des Betriebssystems, sowie Adressen von Webseiten der Raspberry Pi Foundation, die sich auf das Betriebssystem beziehen.

/etc/papersize enthält den Namen der Papiergröße (`a4`), die standardmäßig verwendet werden soll, falls die Umgebungsvariablen PAPERSIZE und PAPERCONF leer sind

/etc/timezone informiert über die Zeitzone, die im Betriebssystem festgelegt ist, zum Beispiel `Europe/Berlin`. Diese Datei dient aber nicht zur Änderung der Zeitzone.

Verzeichnis /proc

/proc/cpuinfo enthält Angaben zum Modell des Raspberry Pi (model), zum Typ des SoC (hardware und revision) und zu den Prozessorkernen der CPU. Außerdem findet man dort die individuelle Seriennummer des Geräts (serial).

/proc/device-tree/model enthält die Modellbezeichnung des Geräts

/proc/device-tree/serial-number enthält die individuelle Seriennummer des Geräts

Verzeichnis /sys

/sys/class/graphics/fb0/virtual_size enthält die Bildschirmauflösung Breite,Höhe

/sys/class/net/eth0/address enthält die MAC Adresse von eth0

Verzeichnis /usr

/usr/share/doc ist ein Verzeichnis mit zahlreichen Dokumentationsdateien

Verzeichnis /var

/var/lib/alsa/asound.state speichert die Einstellung der Lautstärke von Audioausgabegeräten vor dem Herunterfahren des Betriebssystems (siehe Seite 67)

/var/mail/pi enthält mails des Betriebssystems (zum Beispiel von `at`, siehe Seite 115)

2.1.7 Systemsicherung und Klonen der Speicherkarte

SD-Speicherkarten (SD- oder microSD-Card) haben eine begrenzte Lebensdauer. Es ist daher sinnvoll, eine Sicherungskopie (Backup) auf einer Festplatte zu speichern und bei Bedarf auf eine neue SD-Speicherkarte zu übertragen (Klonen).

Wir gehen davon aus, dass ein Rechner mit Linux-System und ausreichend freiem Speicher auf einer Festplatte (siehe unten bei c), sowie mit einem Gerät zum Lesen und Schreiben von SD-Speicherkarten (SD- oder microSD-Card) zur Verfügung steht. Wir verwenden das Terminal.

SD-Speicherkarte ⟶ Festplatte

Wir ermitteln zunächst den Namen der Speicherkarte im Betriebssystem.

a) Ohne eingelegte Speicherkarte geben wir den Befehl

```
lsblk -p
```

ein, um die Speichermedien des Rechners aufzulisten. Nun stecken wir die zu kopierende Speicherkarte (Quelle) in das Lesegerät des Rechners und führen obigen Befehl erneut aus. Das zusätzlich angezeigte Gerät (hier mit `X` abgekürzt) ist die Speicherkarte. Ist sie partitioniert, werden darunter auch die Partitionen (hier mit `Y` abgekürzt) gezeigt.

b) In der rechten Spalte der Auflistung steht der „mountpoint“ für die eingehängten Partitionen. Falls Partitionen der Speicherkarte eingehängt sind, werden sie der Reihe nach mit dem Befehl

```
umount Y
```

ausgehängt, wobei `Y` der Name der Partition ist. Wir hängen aber nur die Partitionen der zu beschreibenden Speicherkarte aus!

c) Nun geben wir mit Administratorrechten den Befehl zum Kopieren des Inhalts der Speicherkarte in den Rechnerspeicher.

```
sudo dd bs=4M if=/dev/X of=datei conv=fsync
```

Dabei ist `X` der Name der Speicherkarte (nicht der Name einer Partition). `datei` ist ein beliebiger Name der Imagedatei, die wir schreiben. Man könnte sie `pios_YYMMDD` nennen, wobei `YY` die beiden letzten Ziffern der aktuellen Jahreszahl, `MM` der Monat als zweistellige Zahl (gegebenenfalls mit führender 0) und `DD` der Tag des Monats als zweistellige Zahl (gegebenenfalls mit führender 0) sind. Man beachte, dass der Speicherbedarf dieser Datei genauso groß ist wie die Größe der Speicherkarte.

Der Schreibvorgang dauert ein paar Minuten (abhängig von der Größe der Speicherkarte)! Um sicherzustellen, dass der Kopiervorgang vollständig beendet wird, geben wir anschließend den Befehl

```
sync
```

ein. Danach entnehmen wir die Speicherkarte (Quelle).

Festplatte ⟶ SD-Speicherkarte

Wir brauchen eine leere Speicherkarte (oder eine, deren Daten nicht mehr gebraucht werden), die mindestens so groß ist, wie die Speicherkarte, die wir kopieren (klonen) wollen. Am besten ist es, wenn die zu kopierende Speicherkarte und die leere Speicherkarte aus der gleichen Serie des gleichen Herstellers stammen. Ansonsten kann es Probleme beim Beschreiben der leeren Karte geben, es kann aber auch alles gut gehen.

Wir legen die zu beschreibende Speicherkarte (Ziel) in das Lesegerät des Rechners. Der Speicher muss mindestens so groß sein wie die Imagedatei `datei`. Warnung: Daten, die sich noch auf dieser Speicherkarte befinden, werden überschrieben!

Den Namen der Speicherkarte im Betriebssystem ermitteln wir wie oben (Seite 65). Wir beschreiben die Speicherkarte (Ziel) mit folgendem Befehl. Prüfen Sie aber vorher nochmals, dass `X` die zu beschreibende Speicherkarte ist (damit sie nicht irrtümlich Ihre Festplatte überschreiben)!

```
sudo dd bs=4M if=datei of=/dev/X conv=fsync
```

Der Schreibvorgang dauert ein paar Minuten (abhängig von der Größe der Speicherkarte)! Um sicherzustellen, dass der Kopiervorgang vollständig beendet wird, geben wir anschließend den Befehl

```
sync
```

ein. Danach entnehmen wir die Speicherkarte (Ziel) und sind fertig.

2.2 Audio

Schall kann durch musikalische Noten oder durch analoge physikalische Größen beschrieben werden. *MIDI* ist ein technische Vorschrift (englisch: technical standard), welche die musikalische Beschreibung von Schall auf elektronische Geräte überträgt.

PCM (pulse-code modulation) ist die Standardmethode zur Übersetzung analoger physikalischer Schallgrößen in digitale Daten, welche im Speicher eines Computers oder auf Datenträgern als digitale Daten vorliegen.

Die analogen Größen werden mit einer bestimmten *Abtastrate*, englisch *sampling frequency*, digitalisiert. Übliche Abtastraten sind 44,1 kHz (in vielen Audiogeräten und in CDs) und 48 kHz (in DVDs). Größere Abtastraten erhöhen die Datenmenge, die pro Zeit verarbeitet werden muss. Für Sprachaufnahmen mit dem Raspberry Pi sind Mikrophone mit einer Abtastrate von nur 16 kHz geeignet. Damit ist der Klang bei hohen Tönen eingeschränkt, aber die Datenrate ist leichter zu bewältigen. Die Datenrate hängt nicht nur von der Abtastrate, sondern auch von der Anzahl der Kanäle (1 für mono, 2 für stereo und möglicherweise mehr) und der Abtasttiefe ab. Die Abtasttiefe (audio bit depth) ist die Anzahl der Bits für einen abgetasteten Wert. Bei linearem PCM (LPCM) sind 8 bit bis 24 bit üblich, zum Beispiel 16 bit für Audio CDs.

Ein *Datenformat* ist eine Vorschrift für die Anordnung von bestimmten Daten. Ein *Audioformat* ist ein Datenformat für digitale Daten in einer Datei oder in einem Datenstrom, die ein Schallereignis beschreiben. Audiogeräte erzeugen oft große Datenmengen in einem einfachen Audioformat. Die Daten einer Audiodatei können durch mathematische Verfahren in ein anderes Audioformat gewandelt werden, das weniger Speicherplatz beansprucht. Dieser Vorgang wird Datenkompression oder kurz *Kompression* genannt. Die Umkehrung heißt *Dekompression*. Die Kompression von Audiodaten kann ohne oder mit Informationsverlust geschehen. Es gibt zahlreiche verschiedene Audioformate. Die eben genannten Begriffe lassen sich sinngemäß auf Videodaten übertragen.

Ein Gerät oder ein Algorithmus zur Kompression wird englisch *encoder* genannt (oder coder), eines zur Dekompression *decoder*. Beides zusammen ist ein *codec*. Ein *audio codec* ist ein codec für Audiodaten und ein *video codec* ist ein codec für Videodaten.

In einem Rechner mit Linux-Betriebssystem werden Geräte zur Audio-Ein- und Ausgabe vom Kernel gesteuert. Dies geschieht über spezielle Computerprogramme, die *Gerätetreiber* (kurz Treiber) heißen und Teil des Kernels sind. Die Sammlung aller Audiogerätetreiber des Kernels nennen wir *Treibersystem*. Es gibt zwei Treibersysteme, ALSA und OSS. Das ältere OSS (Open Sound System) wird nur noch selten eingesetzt und wir gehen nicht weiter darauf ein.

2.2.1 ALSA

Fast alle Linux-Kernels, einschließlich der vom Raspberry Pi OS, enthalten das Audiotreibersystem ALSA (Advanced Linux Sound Architecture). Der Zugang zu den ALSA Gerätetreibern erfolgt über das *ALSA kernel interface* (= ALSA kernel API).

sound server

Programme, die Audiogeräte benutzen, greifen in der Regel nicht direkt auf das ALSA kernel interface zu, weil das schwierig ist. Sie nutzen einen *sound server*. Ein sound server ist ein Programm, das zwischen ALSA kernel interface und Anwenderprogrammen sitzt. Es gibt verschiedene sound server, zum Beispiel ALSA, Pipewire, PulseAudio und JACK.

Man beachte, dass ALSA zwei Teile hat: 1) das ALSA kernel interface und 2) den ALSA sound server im user space, der die ALSA userspace library `alsa-lib` und Programme wie `aplay`, `arecord`, `alsamixer` und `amixer` enthält. Das ALSA kernel interface wird immer gebraucht. Der ALSA sound server kann ersetzt werden.

Der ALSA sound server hat einige Beschränkungen. Pipewire und PulseAudio arbeiten mit manchen Anwenderprogrammen besser zusammen. Der Firefox Browser zum Beispiel arbeitet garnicht mit dem ALSA sound server, aber wir können den Firefox Browser ohne X-System sowieso nicht verwenden. Andererseits ist der ALSA sound server schlank und kann gut von Shellscripten und Programmen angesprochen werden.

Die Grundeinstellungen für den ALSA sound server sind in der Konfigurationsdatei `/usr/share/alsa/alsa.conf` festgelegt, welche unter anderem die Datei `~/.asoundrc` (im Home-Verzeichnis ~ des Nutzers) lädt, wenn es sie gibt. Änderungen der Standardeinstellungen werden wir nur in der Konfigurationsdatei `~/.asoundrc` vornehmen.

Die Lautstärke von Audioausgabegeräten wird beim Herunterfahren (shutdown) des Betriebssystems gespeichert und beim Starten wieder verwendet.

Leider arbeitet der display driver `vc4-kms-v3d` nicht mit dem ALSA sound server zusammen, siehe Seite 7. Dies Problem wird durch den Ersatz von `vc4-kms-v3d` durch `vc4-fkms-v3d` (mit einem `f` wie fake) gelöst, siehe (siehe Seite 46).

ALSA-Soundkarten

Jedes Audiogeräte, das am Rechner angeschlossen ist, wird von ALSA als (mindestens) eine ALSA-Soundkarte beschrieben. Der Begriff ALSA-Soundkarte, kurz card, hat für den ALSA sound server eine besondere Bedeutung, unabhängig davon, was man sonst unter „Soundkarte“ verstehen mag. Jede ALSA-Soundkarte kann mehrere ALSA-Geräte

(devices) enthalten. Diese darf man nicht mit Betriebssystemgeräten verwechseln: Das Wort „Gerät“ oder „device“ ist auch doppeldeutig. Außerdem kann jedes ALSA-Gerät noch Sub-Geräte (subdevices) enthalten., was uns aber oft nicht interessiert. Ohne Angabe eines Sub-Geräts werden alle Sub-Geräte benutzt.

aplay und arecord

ALSA enthält die Programme `aplay` zum Abspielen und `arecord` zur Aufnahme von Audiosignalen. Beide können mit wenigen Formaten von Audiodateien umgehen, nämlich Au (Dateiendung `.au` oder `.snd`), raw (Dateiendung `.pcm`, `.raw` oder `sam` oder ohne Dateiendung), VOC (Dateiendung `.voc`) und WAVE (Dateiendung `.wav`). Wird kein Audioformat angegeben, nehmen beide Programme WAVE. Das Audio-Dateiformat MP3 können diese Programme nicht verarbeiten. `aplay` und `arecord` sind auch wichtige Werkzeuge, um Information über die angeschlossenen Audiogeräte zu bekommen.

Wichtige Optionen der beiden Programme sind

-c x Angabe der Anzahl der Audio-Kanäle (`1` – `32`), Standard ist `1` (mono)
-i die Leertaste der Tastatur beginnt oder endet eine Pause bei Aufnahme / Abspiel
-d x Dauer in s, Standard ist `0` unendlich (bis Abbruch, zum Beispiel mit `Ctrl-C`)
-D x Angabe von Plug-In (und Audiogerät) für Aufnahme oder Abspiel
-f x Abtasttiefe und Format des Werts, Standard `U8` (8 bit Ganzzahl ohne Vorzeichen)
-l Liste von Audiogeräten (cards, devices) für Aufnahme beziehungsweise Abspiel
-L Liste von Plug-ins (Unterprogramme von ALSA), die bei ALSA PCMs heißen
-N sofortiges Ende, wenn das Audiogerät nicht verfügbar ist, Standard ist Warten
-q Unterdückung der Ausgabe von Textmeldungen auf dem Bildschirm (quiet)
-r x Angabe der Abtastrate in Hz oder kHz (wenn < 300), Standard ist `8000`
-t x Angabe des Audio-Dateiformat (`voc`, `wav`, `raw` oder `au`), Standard ist `wav`

Ausgabelautstärke oder Eingangsempfindlichkeit: alsamixer und amixer

Für alle ALSA-Soundkarten zur Audioausgabe kann die Lautstärke mit dem Programm `alsamixer` festgelegt werden. Es ist im Paket `alsa-utils` enthalten, das bereits installiert sein sollte und ansonsten durch den Befehl

```
sudo apt-get install alsa-utils
```

installiert wird. Das Programm `alsamixer` wird mit dem Befehl

```
alsamixer
```

aufgerufen und hat eine graphische Oberfläche (ncurses user interface). Nach Drücken der Taste F6 kann man eine ALSA-Soundkarte auswählen, für welche die Lautstärke eingestellt werden soll. Die Festlegung, welche ALSA-Soundkarte als Standard für die Audioausgabe dient, wird dadurch nicht verändert. Die Einstellung der Lautstärke erfolgt mit den Pfeiltasten ↑ und ↓, die einen virtuellen Schieberegler bewegen.

Manche ALSA-Soundkarten dienen nicht der Audioausgabe (in alsamixer Playback genannt), sondern der Eingabe, steuern also ein Mikrophon (in alsamixer Capture ge-

nannt). Und es gibt ALSA-Soundkarten, die sowohl Audioausgabe als auch Audioeingabe steuern, zum Beispiel über eine USB-Soundkarte. Die Eingangsempfindlichkeit (Verstärkung) einer Audioeingabe wird ähnlich wie die Lautstärke einer Audioausgabe mit alsamixer eingestellt. Das Programm ist leicht zu bedienen. Mit der Escape-Taste ESC kann man das Programm verlassen.

Eine Alternative zu `alsamixer` ist das Programm `amixer`, das ebenfalls im Paket `alsa-utils` enthalten ist. Es kann mit einzeiligen Befehlen arbeiten und hat keine graphische Oberfläche. Damit kann es auch in Shellscripten und Programmen verwendet werden. Die Option `-c` mit einer Nummer erlaubt die Auswahl einer ALSA-Soundkarte (card). Danach folgt eine Anweisung für diese ALSA-Soundkarte. Mit dem Befehl

```
amixer -c 0 scontents
```

erhält man eine Übersicht der Steuerungselemente (simple mixer controls) der ALSA-Soundkarte. Für die Steuerungselemente kann man einen Wert setzen. Bei der Lautstärke wird normalerweise ein Pegel in der Einheit 0,01 dB gesetzt. Zum Beispiel wird mit

```
amixer -q -c 0 -M sset "PCM" -- -1000
```

die Lautstärke für die ALSA-Soundkarte 0 auf −10 dB gesetzt. Dabei unterdrückt die Option `-q` die Textausgabe über den `amixer`-Zustand auf dem Bildschirm. Die Option `-M` sorgt dafür, dass Prozentangaben (siehe unten) von `amixer` in gleicher Weise berechnet werden, wie es `alsamixer` macht. Die in (einfache oder doppelte) Anführungsstriche gesetzte Zeichenkette gibt das Steuerungselement an, was gesetzt werden soll, wobei "PCM" die Lautstärke meint. Pegeländerungen erreicht man, indem an den Zahlenwert ein Minuszeichen oder Pluszeichen angefügt wird.

```
amixer -q -c 0 -M sset "PCM" -- 300+
```

hebt den Lautstärkepegel um 3 dB auf −7 dB an. Statt als Pegelwert kann die Lautstärke auch als Prozentwert zwischen 0 % und 100 % gesetzt werden. Mit

```
amixer -q -c 0 -M sset "PCM" 100%
```

wird die Lautstärke für die ALSA-Soundkarte 0 auf 100 % gesetzt und mit kleineren Prozentzahlen kann man die Ausgabe leiser machen. Bei `100%` kann es schon zu Übersteuerungen (verzerrter Klang) kommen und man sollte dann etwas weniger nehmen, zum Beispiel 98 %. Auch Änderungen sin in Prozent möglich. Mit

```
amixer -q -c 0 -M sset "PCM" 10%-
```

wird die Lautsärke um 10 % von 100 % auf 90 % gesenkt.

Die Audioausgabe der ALSA-Soundkarte 0 wird mit dem Befehl

```
amixer -c 0 sset "PCM" mute
```

stumm geschaltet und der Befehl

```
amixer -c 0 sset "PCM" unmute
```

hebt die Stummschaltung wieder auf.

Entsprechend kann man die Eingangsempfindlichkeit für die Audioeingabe mit alsamixer oder amixer setzen. Zum Beispiel für ein Mikrophon, das an USB als angeschlossen ist und die ALSA-Soundhartennummer 2 hat:

```
amixer -c 2 sset "Mic" 45%
```

Statt einer Prozentangabe kann man am Ende der Befehlszeile auch einen Capture-Wert als Zahl angeben. Ohne die Option `-q` erhält man Information zur Einstellung, unter anderem auch den Wertebereich, den man verwenden sollte, sowie die Umrechnung von Capture-Werten in Prozentwerte. Um die Audioeingabe der ALSA-Soundkarte 2 stumm zu schalten, gibt man den Befehl

```
amixer -c 2 sset "Mic" uncap
```

und der Befehl

```
amixer -c 2 sset "Mic" cap
```

hebt die Stummschaltung wieder auf.

Allerdings beachten einige Programme die Vorgaben des Betriebssystems nicht, die über `alsamixer` oder `amixer` gegeben wurden, und verwenden eigene Steuerbefehle.

2.2.2 HDMI, Analogausgang und USB (Ein-und Ausgabe)

In den Audio- und Video-Beispielen dieses Manuskripts (zum Beispiel im Abschnitt 6.1.7 ab Seite 197) gehen wir in der Regel davon aus, dass ein HDMI-Monitor mit Lautsprechern angeschlossen ist und am analogen Audioausgang ein Audio-Empfangsgerät (im einfachsten Fall ein 50 Ω-Lautsprecher, siehe Seite 17) angeschlossen werden kann. „Kann“ bedeutet: Der Raspberry Pi erkennt nicht, ob am analogen Audioausgang ein Audio-Empfangsgerät angeschlossen ist.

Unten (siehe Seite 73) behandeln wir den Anschluss einer USB-Soundkarte und wer eine solche benutzt (was durchaus zu empfehlen ist), muss in einigen späteren Beispielen die ALSA-Soundkartennnummer und die ALSA-Gerätenummer anpassen.

Prüfung der Audioausgabe über HDMI und Analogausgang

Wir gehen zunächst davon aus, dass als Audioausgabegeräte ein Lautsprecher (mit oder ohne Verstärker) am analogen Audioausgang und ein Monitor mit Lautsprechern am HDMI-Ausgang angeschlossen sind, aber keine USB-Soundkarte angeschlossen ist.

Information zu den verfügbaren ALSA-Soundkarten (cards) erhalten wir mit

```
cat /proc/asound/cards
```

Wir erhalten eine Liste, in der jede ALSA-Soundkarte eine Nummer und einen Namen hat. Die Nummer steht am Zeilenanfang und der Name (gefolgt von Leerzeichen) danach in eckigen Klammern. Nach einem Doppelpunkt folgt eine kurze Erklärung. Zum Beispiel könnte der erste HDMI Audioausgang die Nummer 0 und den Namen `b1` haben und der analoge Audioasusgang die Nummer 1 und den Namen `Headphones`. Wenn es

einen zweiten HDMI Audioausgang oder eine USB-Soundkarte gibt, wird die Ausgabe natürlich anders sein. Man kann ALSA-Soundkarten über ihre Nummer oder ihren Namen ansprechen. (Erinnerung: Die Bezeichnung „Soundkarte“ ist mehrdeutig.)

Genauere Information zu cards und devices für die die Audio-Ausgabe erhalten wir mit

```
aplay -l
```

Zum Beispiel könnte der erste HDMI-Ausgang als ALSA-Soundkarte 0, kurz card 0, device (ALSA-Gerät) 0 angezeigt werden und der analoge Audioausgang als ALSA-Soundkarte 1, kurz card 1, device (ALSA-Gerät) 0. Die aufgeführten Sub-Geräte interessieren uns hier nicht.

Wir benutzen das Programm `aplay`, um die Audioausgabe in einfacher Weise zu testen. Zunächst prüfen wir card 0 mit dem Befehl

```
aplay -D hw:0,0 ~/.bashrc
```

und sollten ein kurzes Rauschen hören. `~/.bashrc` wurde willkürlich ausgewählt, um irgendeine Datei im Befehl anzugeben. Sie enthält keine Audiodaten und deshalb hört man nur Rauschen. In gleicher Weise können wir auch card 1 prüfen:

```
aplay -D hw:1,0 ~/.bashrc
```

Wahl des Standard-Geräts für die Audioausgabe

Im einfachsten Fall erfolgt die Audioausgabe über HDMI (durch die ALSA-Soundkarte 0) oder über den analogen Ausgang, also die 3,5 mm-Buchse , siehe Seite 13) (durch die ALSA-Soundkarte 1). Voreingestellt (Standard) ist die Ausgabe über HDMI. Wenn man eine USB-Soundkarte hat, kann auch diese für die Audioausgabe benutzt werden (durch die ALSA-Soundkarte 2, siehe Seite 73).

Welche ALSA-Soundkarte als Standardausgabe dient, wird in der Datei `~/.asoundrc` festgelegt. Durch `raspi-config` wird in zwei Zeilen dieser Datei entweder `card 0` oder `card 1` oder `card 2` geschrieben. Wir ändern die Datei aber nicht direkt, sondern sicherer mithilfe von `raspi-config`.

Das Konfigurationsprogramm kann ein Audioausgabegerät als Standard setzen:

```
sudo raspi-config
```

Man wählt unter `1 System Options` und danach S2 Audio die gewünschte Audioausgabe (audio output). Für den Analogausgang ist dies `...Headphones`. Dann kann man `raspi-config` verlassen (Finish) und die Audioausgabe mit

```
aplay ~/.bashrc
```

prüfen. Anders als oben (auf Seite 70) geben wir im Befehl nicht an, wo die Audio-Ausgabe stattfinden soll und es wird daher die Standardausgabe verwendet.

Eine Alternative ist der Befehl

```
speaker-test
```

der ein Rauschen ausgibt. Es wid mit `Ctrl-C` beendet.

Eingabe durch USB-Mikrophon

Eine 3,5 mm-Buchse zur Eingabe analoger Signale fehlt beim Raspberry Pi. Man kann aber über USB ein Mikrophon anschließen. Um zu prüfen, ob das Mikrophon vom Betriebssystem erkannt wird, führen wir vor und nach Anstecken des Mikrophons den Befehl

```
lsusb
```

aus. Nach dem Anschluss sollte eine Zeile mehr ausgegeben werden, in der das angeschlossene Gerät genannt wird. Der Anschluss an den Raspbery Pi wird durch Busnummer (hinter „Bus“) und Gerätenummer (hinter „Device“) beschrieben. Achtung: Wenn man das Gerät entfernt und dann wieder ansteckt, kann sich die Gerätenummer geändert haben. Nach „ID“ folgt eine Zeichenkette (ID), in der die Zahl vor dem Doppelpunkt (Vendor-ID) den Hersteller kennzeichnet. Nach dem Doppelpunkt folgt eine Zahl (Product-ID), die das Produkt des Herstellers kennzeichnet. Nach der ID sollte eine verständliche Kurzbeschreibung des Geräts stehen. Allerdings kann sie auch fehlen.

Wenn das Mikrophon vom Betriebssystem erkannt wurde, prüfen wir, ob ALSA es als ALSA-Soundkarte erkannt hat und wie es bezeichnet wird. Wir führen vor und nach Anstecken des Mikrophons den Befehl

```
cat /proc/asound/cards
```

aus. Der Listeneintrag, welcher nach dem Anschluss hinzugekommen ist, beschreibt das Mikrophon. Die erste Zahl ist die ALSA-Soundkartennummer. HDMI-Ausgang und analoger Audio-Ausgang haben in der Regel die Soundkartennummern `0` und `1`. Das Mikrophon sollte daher die ALSA-Soundkartennummer `2` haben, wenn keine weitereren Audiogeräte angeschlossen wurden.

Information zur ALSA-Soundkarte, welche dem angeschlossenen USB-Mikrophon zugeordnet ist, erhält man mit dem Befehl

```
arecord -l
```

In diesem Beispiel hat das Mikrophon die ALSA-Soundkartennummer `2` und diese ALSA-Soundkarte enthält nur ein ALSA-Gerät (device) mit der Nummer `0` und dieses hat nur ein Sub-Gerät (subdevice) mit der Nummer `0`. Mit der ALSA-Soundkartennummer und der ALSA-Gerätenummer können wir das Mikrophon im ALSA-System steuern.

Um das Mikrophon zu prüfen, nehmen wir 5 Sekunden lang unsere Stimme auf und speichern sie (im Format WAV) in einer Datei namens `mt`. Das geschieht mit dem Befehl

```
arecord -f cd -d 5 -D hw:2,0 mt
```

Mit `hw:2,0` wird das Mikrophon ausgewählt, denn es hat (in diesem Beispiel) die Soundkartennummer `2` und die ALSA-Gerätenummer `0`. Dann spielen wir mit dem Befehl

```
aplay mt
```

die Aufnahme auf dem Standard-Audioausgabegerät ab und sollten es hören können.

Ein- und Ausgabe über USB-Soundkarte

Die Ein- und Ausgabe von Audiosignalen kann auch über eine USB-Soundkarte erfolgen. Um zu prüfen, ob die USB-Soundkarte vom Betriebssystem erkannt wird, führen wir vor und nach dem Anstecken an einer USB-Buchse den Befehl

```
lsusb
```

aus. Wie beim USB-Mikrophon (Seite 72) sollte nach dem Anschluss eine Zeile mehr ausgegeben werden, in der das angeschlossene Gerät genannt wird (siehe Seite 72).

Wenn die USB-Soundkarte vom Betriebssystem erkannt wurde, prüfen wir, ob sie auch als ALSA-Soundkarte erkannt wird und führen vor und nach dem Anstecken den Befehl

```
cat /proc/asound/cards
```

aus. Der Listeneintrag, der nach dem Anschluss hinzugekommen ist, beschreibt die USB-Soundkarte als ALSA-Soundkarte. Diese kann durch die ALSA-Soundkartennummer adressiert werden. HDMI-Ausgang und analoger Audio-Ausgang haben in der Regel die ALSA-Soundkartennummern `0` und `1`. Die USB-Soundkarte sollte also die ALSA-Soundkartennummer `2` haben, wenn keine anderen Audiogeräte angeschlossen wurden.

Welche ALSA-Soundkarten für die Audio-Ausgabe geeignet sind, erfahren wir mit

```
aplay -l
```

In diesem Beispiel gibt es drei ALSA-Soundkarten für die Audio-Ausgabe, nämlich ALSA-Soundkarte 0 (`HDMI`) mit ALSA-Gerät 0, ALSA-Soundkarte 1 (`Headphones`) mit ALSA-Gerät 0 und ALSA-Soundkarte 2 (`Device`) mit ALSA-Gerät 0 .

Die USB-Soundkarte hat hier also die ALSA-Soundkartennummer `2` und sie enthält nur ein ALSA-Gerät (device) mit der Nummer `0`. Die ALSA-Soundkarte`2` hat hier leider den blödsinnigen Namen `Device` (bitte nicht verwechseln mit den anderen Bedeutungen von „device“). Ob ein Mikrophon oder ein Kopfhörer angesteckt sind, weiß ALSA nicht.

Wir schließen nun einen Kopfhörer an den Kopfhörerausgang der USB-Soundkarte an. Wenn der Mediaplayer `mplayer` installiert wurde (siehe Seite 186), können wir eine Audiodatei `x.mp3`, die im aktuellen Verzeichnis liegt, durch folgenden Befehl abspielen:

```
mplayer -ao alsa:device=hw=2.0 x.mp3
```

Der entsprechende Befehl für den Mediaplayer `mpv` (siehe Seite 184) ist

```
mpv --audio-device=alsa/plughw:2,0 x.mp3
```

und für den Mediaplayer `vlc` mit dem Schnittstellenmodul `rc` (siehe Seite 193) ist es

```
vlc --intf rc --aout alsa --alsa-audio-device=hw:2,0 x.mp3
```

Natürlich kann man auch die USB-Soundkarte als Standard-Audioausgabegerät wählen, siehe Seite 71.

Wir prüfen nun, welche ALSA-Soundkarten für die Audioeingabe geeignet sind:

```
arecord -l
```

In diesem Beispiel gibt es nur die von ALSA als `Device` bezeichnete USB-Soundkarte.

Sie hat die ALSA-Soundkartennummer `2` und es gibt nur ein ALSA-Gerät (device) mit der Nummer `0` . Ob ein Mikrophon mit der USB-Soundkarte verbunden ist, oder nicht, können wir mit obigem Befehl nicht erkennen.

Wir schließen nun ein Mikrophon an den Mikrophoneingang der USB-Soundkarte an. Mit der ALSA-Soundkartennummer und der ALSA-Gerätenummer können wir das Mikrophon im ALSA-System steuern. Zu Prüfung nehmen wir 5 Sekunden lang unsere Stimme auf und speichern sie (im Format WAV) in einer Datei namens `mt`:

```
arecord -f cd -d 5 -D hw:2,0 mt
```

Mit der ALSA-Soundkartennummer `2` und der ALSA-Gerätenummer `0` wird hier das Mikrophon ausgewählt.

Falls die Fehlermeldung „Channels count non available“ („Kanalanzahl nicht unterstützt“) erscheint, müssen wir noch die richtige Kanalanzahl als Option angeben. Mit einem einkanaligen Billigmikrophon (Mono) brauchen wir die Option `-c 1` .

```
arecord -f cd -c 1 -d 5 -D hw:2,0 mt
```

Dann spielen wir die Aufnahme mit dem Befehl

```
aplay mt
```

auf dem Standard-Audioausgabegerät ab und sollten unsere Stimme hören. Falls es zu leise klingt, hebt man die Eingangsempfindlichkeit oder die Ausgabelautstärke an.

2.3 Autostart

Es ist oft nützlich, ein eigenes Skript oder Programm nach dem Systemstart automatisch ausführen zu lassen (Autostart).

2.3.1 Autostart mit .bashrc oder rc.local

.bashrc Wie oben beschrieben (siehe Seite 50), wird beim Starten der Shell (Bash) die versteckte Datei `.bashrc` gelesen. Man kann, am Ende dieser Datei, den Pfad zu einer ausführbaren Datei setzen, zum Beispiel einem Shellscript. Dieses wird dann mit dem Start des Terminals für den angemeldeten Nutzer ausgeführt.

rc.local (veraltet) Früher war es in Linux Betriebssystemen üblich, für den Autostart eine Datei namens `rc.local` im Verzeichnis `/etc` zu verwenden. Es genügte, den Pfad zum eigenen Programm in `rc.local` einzutragen. Der Eintrag musste vor der Zeile mit `exit 0` stehen, welche am Ende der Datei bleibt. Das Programm wurde mit Root-Rechten ausgeführt, ohne dass `sudo` vorangestellt werden musste. Das Programm wurde vor dem Start des X-Systems ausgeführt, so dass die grafische Oberfläche dem Programm nicht zur Verfügung stand. Wenn der Systemstart während der Programmausführung weiterlaufen sollte, wurde ein & am Ende des Eintrags gesetzt (Hintergrundprozess).

Probleme waren möglich aufgrund der fehlenden Abstimmung mit dem Start von Daemonen während der Initialisierung des Systems. Diese Methode ist veraltet, seit `systemd` der Daemon für den Systemstart ist.

2.3.2 Dienstanmeldung in systemd

Der Systemstart wird vom Daemon `systemd` durchgeführt. Zum Autostart eines eigenen Programms, zu Beispiel des Pythonprogramms `meinprog.py`, dessen Datei im Verzeichnis `/home/ahg/progs` liegt, wird eine Datei `meinprog.service` mit folgendem Inhalt geschrieben.

```
[Unit]
Description=Kurzbeschreibung von meinprog.py
After=network.target

[Service]
ExecStart=/usr/bin/python3 -u meinprog.py
WorkingDirectory=/home/ahg/progs
StandardOutput=inherit
StandardError=inherit
Restart=always
User=pi

[Install]
WantedBy=multi-user.target
```

Diese Datei ist eine Konfigurationsdatei und wird nicht ausführbar gemacht. Sie wird nur in das Verzeichnis `/etc/systemd/system` verschoben:

```
sudo mv meinprog.service /etc/systemd/system
```

Um den Autostart zu testen startet man das Programm mit dem Befehl

```
sudo systemctl start meinprog.service
```

und kann dann den Zustand des Programms mit

```
sudo systemctl status meinprog.service
```

abfragen. Mit

```
sudo systemctl stop meinprog.service
```

wird der Test beendet. Funktioniert es, kann man mit

```
sudo systemctl enable meinprog.service
```

den Autostart als Dienst bei `systemd` anmelden. Die mit `After` beginnende Zeile stellt sicher, dass unser Programm erst nach Bereitstellung des Netzwerks ausgeführt wird.

Statt eines Pythonprogramms kann auch ein Shellscript gestartet werden.

3 Administration des Systems

3.1 Paketverwaltung, Netzwerk-Analyse und Hilfe

Programme werden in Form von „Paketen“ gespeichert, die von der Paketverwaltung (Debian Package Manager) verwaltet werden. Für den Nutzer meistens unsichtbar, arbeitet das Programm `dpkg` im Hintergrund der Paketverwaltung. `dpkg` arbeitet mit Paketdateien, welche die Endung `deb` haben. Mehr Funktionen und eine einfachere Schnittstelle zum Nutzer bietet Programme des Paketverwaltungssystems APT, welches auf auf `dpkg` aufbaut. Die für Raspbian verfügbaren Pakete werden in einer Datenbank geführt, die auf dem Rechner gespeichert ist.

Das Paketverwaltungssystem APT (Advanced Packaging Tool), das es schon seit 1998 gibt, enthält verschiedene Programme, die über die Kommandozeile aufgerufen werden können. Dazu gehören `apt-get` und `apt-cache`. Ein neueres Programm (seit 2014), welches die eben genannten teilweise ersetzen kann, ist `apt`. Beim direkten Aufruf durch den Nutzer ist `apt` meistens besser geeignet, weil die Befehle kürzer sind und zum Beispiel den Fortschritt von länger dauernden Aktionen darstellen. Für den Aufruf aus einem anderen Programm, zum Beispiel einem Bash-Skript, ist `apt` dagegen weniger geeignet, da die nutzerfreundliche Ausgabe von `apt` die automatische Verarbeitung erschwert.

Eine Sammlung von Programmen, auf die verschiedene Nutzer zugreifen können, heißt Software repository oder kurz Repository. Im Internet gibt es verschiedene Repositories, von denen man Pakete für unser Betriebssystem herunterladen kann, in der Regel mithilfe des Paketverwaltungssystems.

Tabelle 3.1 zeigt eine Liste grundlegender Befehle für die Paketverwaltung.

Funktion	mit apt-get / apt-cache	mit apt
Aktualisierung der Datenbank	`sudo apt-get update`	`sudo apt update`
Aktualisierung der Pakete	`sudo apt-get upgrade`	`sudo apt upgrade`
Installation des Pakets *x*	`sudo apt-get install x`	`sudo apt install x`
Löschung des Pakets *x*	`sudo apt-get remove x`	`sudo apt remove x`
Löschung mit Konfiguration	`sudo apt-get purge x`	`sudo apt purge x`
Automatisches Aufräumen	`sudo apt-get autoremove`	`sudo apt autoremove`
Suche nach *x* in Datenbank	`apt-cache search x`	`apt search x`
Information zum Paket *x*	`apt-cache show x`	`apt show x`

Tabelle 3.1: Wichtige Befehle des Paketverwaltungssystems

3.1.1 Aktualisierung, Paketverwaltung und Netzwerk-Analyse

System-Aktualisierung

Zur Aktualisierung des Betriebssystems und der installierten Pakete (Update) gibt man nacheinander folgende zwei Befehle ein. Der erste aktualisiert die Paketdatenbank im eigenen Rechner und der zweite holt eine neuere Version installierter Pakete, sofern eine neuere verfügbar ist. Die Abarbeitung der beiden Befehle kann lange dauern und man muss eventuell bestätigen, dass man eine gewisse Datenmenge herunterladen möchte.

```
sudo apt update
sudo apt upgrade
```

Befehle zur Paketverwaltung

Die Paketverwaltung beinhaltet Paketinstallation, -löschung und -information. Ein Paket `pak` , das über die Paketverwaltung verfügbar ist, kann mit dem Befehl

```
sudo apt install pak
```

installiert werden. Man kann auch mehrere Pakete mit einer Befehlszeile installieren, wobei die Paketnamen durch Leerzeichen getrennt werden. Wird der Befehl auf ein bereits installiertes Paket angewandt, wird dieses aktualisiert, falls eine neuere Version verfügbar ist.

Will man ein Paket installieren, das nicht in einem Repository, sondern im aktuellen Arbeitsverzeichnis vorliegt, zum Beispiel als Datei `pak.deb`, nimmt man den Befehl

```
sudo apt install ./pak.deb
```

Wichtig ist die Angabe des volsständigen Pfads zur Paketdatei.

Wird ein Paket `pak` nicht mehr gebraucht, kann man es mit

```
sudo apt purge pak
```

vollständig, das heißt: einschließlich der nicht nutzerspezifischen Konfigurationsdateien, wieder entfernen. Nutzerspezifische Konfigurationsdateien sind solche, die im Verzeichnis `/home`, oder einem Unterverzeichnis davon, stehen. Sie bleiben beim `purge` erhalten und müssen „manuell“ gelöscht werden.

Will man nur die Programmdateien eines Pakets löschen, aber die Konfigurationsdateien behalten, schreibt man `remove` statt `purge`. Das ist sinnvoll, wenn die Konfigurationsdateien vom Nutzer angepasst wurden und bei einer eventuellen Neuinstallation des Pakets wieder verwendet werden sollen.

Bei der Installation eines Pakets x werden of andere Pakete oder Bibliotheken (Sammlungen von Hilfsprogrammen) automatisch mit installiert, weil das angeforderte Paket x diese benötigt. Die bleiben erhalten (verwaisen), wenn das Paket x gelöscht wird. Mit

```
sudo apt autoremove
```

werden verwaiste Pakete und Bibliotheken gelöscht.

Mit dem Befehl

```
apt search adventure
```

kann man Pakete suchen, deren Beschreibung einen Begriff, hier zum Beispiel `adventure`, enthält. Information zu einem bestimmten Paket, hier zum Beispiel `sl`, ergibt

```
apt show sl
```

Zusätzliche Pakete sollten in der Regel durch die Paketverwaltung des Betriebssystems (über `apt` oder `apt-get`) installiert werden. Python-Module, die dort nicht, oder nur in veralteter Form verfügbar sind, können durch das Paketverwaltungsprogramm `pip3` installiert werden, welches über das Internet auf ein repository (digitales Archiv) der Python Software Foundation, namens Python Package Index (PyPI), zugreift. Statt `pip3` kann man kurz `pip` schreiben, da nur noch Python 3 verwendet wird.

Wenn Pakete über zwei verschiedene Paketverwaltungen installiert werden, kann es Probleme geben. Um diese einzugrenzen, kann man systemweite Installationen dem Betriebssystem vorbehalten und `pip3` nur in einem besonderen Verzeichnis erlauben, das vom Rest des Systems isoliert ist. Wenn dieses *python virtual environment* aktiviert ist, übernimmt es die Ausführung von Pythonprogrammen. Dafür kann es die standard library des Betriebssystems nutzen und enthält alle notwendigen Module für ein bestimmtes Pythonprojekt. Dies Verfahren der externally managed environments wird vom Raspberry Pi OS der Version bookworm empfohlen und wird später erläutert (Seite 262). Es ist sicher, aber aufwändig.

Zusätzliche Pakete, mittels apt

Folgende Pakete sollten nach der Konfiguration des Systems mit der Paketverwaltung des Betriebssystems installiert werden, sofern sie nicht schon installiert wurden:

- `pipewire` für Audio/Video, wenn `vc4-kms-v3d` verwendet wird (sieh Seite 7)
- `python3` Programmiersprache Python 3 mit Standardbibliothek
- `python3-dev` Ergänzungen für die Programmierung mit Python 3
- `python3-pip` um Module für Python 3 zu installieren
- `python3-lxml` um XML und HTML it Python 3 zu verarbeiten
- `python3-rpi.gpio` um die GPIO mit Python 3 zu steuern
- `python3-numpy` siehe Seite 309
- `python3-scipy` siehe Seite 319
- `python3-matplotlib` siehe Seite 312
- `python3-pypdf` zur Bearbeitung von PDF-Dokumenten
- `python3-websocket`
- `python3-venv` für die Einrichtung von virtual environments (siehe Seite 262)
- `libatlas-base-dev` wird eventuell für Numpy (siehe unten) gebraucht
- `ffmpeg` für die Verarbeitung von Audio- und Videodaten
- `fonts-dejavu-core` Fonts (Schriften) z.B. für `gnuplot`

- `fonts-freefont-ttf` Fonts (Schriften)
- `fonts-liberation` Fonts (Schriften), in der Regel automatisch installiert
- `fonts-wqy-zenhei` chinesische Fonts (Schriften), installiert: 16,8 MB
- `fonts-wqy-microhei` chinesische Fonts, 5,3 MB, seltene Schriftzeichen fehlen
- `gnuplot-nox` Plotprogramm gnuplot, siehe Seite 250
- `lynx lynx-common` textbasierter Webbrowser Lynx, siehe Seite 207
- `mediainfo` MediaInfo liefert Info über Audio- und Videodateien, siehe Seite 197
- `mpv` MPlayer Mediaplayer für Audio- und Videodateien, siehe Seite 184
- `mplayer` MPlayer Mediaplayer für Audio- und Videodateien, siehe Seite 186
- `vlc` VLC Mediaplayer für Audio- und Videodateien
- `vlc-bin` für den VLC nowendiges Paket (VLC's binaries)
- `vlc-plugin-base` für den VLC nowendiges Paket (specialized VLC plugins)
- `w3m-img` schlanker Webbrowser mit Bildern, siehe Seite 209
- `fbi` Anzeige von Bilddateien und PDF (Befehl `fbgs`), siehe Seiten 173 und 179
- `fim` Anzeige von Bilddateien (Alternative zu `fbi`), siehe Seite 173
- `imagemagick` ImageMagick zur Bildbearbeitung und -erstellung , siehe Seite 181
- `bc` programmierbarer Taschenrechner bc für die Kommandozeile, siehe Seite 224
- `dialog` dialog dient der Erstellung von Menüs für die Shell, siehe Seite 155

Weitere nützliche Pakete sind:

- `texlive` schöne Manuskripte und Präsentationen (umfangreich), siehe Seite 239
- `texlive-lang-german` Unterstützung für deutsche Latex-Dokumente
- `texlive-bibtex-extra` für ein Literatuverzeichnis in Latex-Dokumenten
- `biber` externes Programm für ein Literatuverzeichnis in Latex-Dokumenten
- `fonts-cmu` Fonts (Schriften): Computer Modern Unicode fonts (OpenType)
- `fswebcam` steuert die Bildaufnahme mit Webcams über USB (siehe Seite 177)
- `gpm` einfache Mausunterstützung mit copy & paste (siehe Seite 48)
- `links2` schlanker Webbrowser mit Bildern, siehe Seite 210, benötigt `gpm`
- `ttf-mscorefonts-installer` Microsoft Fonts, Teilersatz durch fonts-liberation
- `x264` wandelt Videodateien ins Format MPEG-4 AVC (Advanced Video Coding)

Zusätzliche Pakete, mittels pip3

Das Verfahren der externally managed environments will das Paketverwaltungsprogramm `pip3` auf python virtual environments beschränken. Will man `pip3` trotzdem überall verwenden, muss man die Beschränkung ausdrücklich aufheben.

Man kann mit Administratorrechten (`sudo pip3 install ...`) Python-Module oder Pakete für alle Nutzer installieren. Das ist aber nicht ratsam.

Besser ist, wenn jeder Nutzer (zum Beispiel der Nutzer `ahg`), Modulpakete mit `pip3` nur für sich selbst (lokal) installiert, wofür man keine Administratorrechte braucht.

`pip3 install --user` *paket* `--break-system-packages`

Dabei werden die Module im Verzeichnis `~/.local/lib/pythonX/site-packages` abgelegt, wobei ~ für das home directory (Heimatverzeichnis, `/home/ahg`) und `X` für die

Versionsnummer stehen. Die Option `--break-system-packages` hebt die oben genannte Beschränkung auf.

Mit der zusätzlichen Option `-update` oder kürzer `-U` wird das angegebene Modul, und auch die davon abhängigen (vom angegebenen Paket verwendeten), aktualisiert.

Zum Deinstallieren eines Moduls ... verwendet man den Befehl

```
pip3 uninstall ...
```

Möglicherweise gibt es die Warnung, dass das Verzeichnis `.local/bin` nicht im Suchpfad liegt. Wenn die Datei `/home/ahg/.profile` ergänzt wurde, wie im Abschnitt „.profile“ beschrieben (siehe Seite 50), wird das Problem mit dem nächsten Start (Booten) des Betriebssystems automatisch gelöst.

Netzwerk-Analyse

Das Programmpaket `netsniff-ng` enthält Dienstprogramme für die Untersuchung des Datenverkehrs im lokalen Netzwerk (Netzwerk-Toolkit). Es hat eine eigene Webseite und eine Projektseite auf GitHub. Es wird mit dem Befehl

```
sudo apt install netsniff-ng
```

installiert. Dazu gehört das gleichnamige Programm `netsniff-ng`, ein „Sniffer“.

3.1.2 Systeminformation, Hilfe zu Programmen und zur bash

Systeminformation

Das empfehlenswerte Programm `inxi` liefert eine übersichtliche Zusammenfassung über Betriebssystem und Hardware (die physischen Komponenten des Rechners, wie CPU, Tastatur, angeschlossene Speicherkarten und USB-Sticks, sowie Netzwerkschnittstellen). Es wird mit dem Befehl

```
sudo apt install inxi
```

installiert. Information zu erhält man dann mit dem Befehl `man inxi`. Durch

```
inxi -F
```

wird das Programm ausgeführt. Die Option `-F` bewirkt eine ausführliche Ausgabe.

Auch Das Programm `lshw` informiert über die Hardware. Es wird mit

```
sudo apt install lshw
```

installiert und mit dem Befehl

```
sudo lshw | less
```

ausgeführt. Die Weiterleitung an den terminal pager `less` (den man durch Drücken der Taste `q` beendet) ermöglicht das Lesen der langen Textausgabe.

Mit dem Programm `neofetch` kann man ansehnlich präsentierte Information zum Betriebssystem abrufen (englisch: to fetch). Es wird mit

```
sudo apt install --fix-missing neofetch
```

installiert und durch den Befehl

```
neofetch
```

gestartet. Auf der Webseite https://github.com/dylanaraps/neofetch/wiki ist eine Dokumentation. `neofetch` ist ein `bash`-Script, das im Verzeichnis `/usr/bin/` liegt. Es läuft auf allen Betriebssystemen, auf denen ein Terminal mit `bash` verfügbar ist.

Die Konfigurationsdatei `~/.config/neofetch/config.conf` für den aktuellen Nutzer (wobei ~ das Heimatverzeichnis des Nutzers ist) wird beim ersten Aufruf von `neofetch` erzeugt. Dort kann man Einstellungen vornehmen.

Den Speicherplatz, den Verzeichnisse und Dateien auf den Speichermedien einnehmen, wird mit dem Programm `ncdu` übersichtlich dargestellt. Es wird mit

```
sudo apt install ncdu
```

installiert und nach dem Befehl

```
ncdu
```

berechnet das Programm den Speicherplatz für das aktuelle Verzeichnis und alle Unterverzeichnisse und die Dateien darin. Will man den Speicherplatz für den ganzen Verzeichnisbaum, also alle Verzeichnisse und Dateien, sehen gibt man den Befehl

```
ncdu /
```

ein. Die Berechnung des Speicherplatzes kann dann etwas dauern – abhängig von der Größe der eingehängten Speichermedien. Wichtige Tastenbefehle sind

? Anzeige einer Liste der Tastenbefehle ein / aus
q quit zurück zur Kommandozeile
↑ in der Liste nach oben
↓ in der Liste nach unten
→ wenn ein Verzeichnis markiert ist: öffne das Verzeichnis
← öffne übergeordnetes Verzeichnis (nachdem man in ein Unterverzeichnis ging)
d (delete) lösche das markierte Objekt (Verzeichnis oder Datei)
i (info) Anzeige von Information zum markierten Objekt ein / aus

Wenn man `ncdu` mit der Option `-r` aufruft, verhindert man die Möglichkeit der (vorschnellen) Löschung eines Objekts durch Drücken der Taste `d` (siehe oben).

Hilfe zu Programmen und bash-Befehlen

Auf Seite 78 wurde gezeigt, wie man Hilfe zu einem Paket `paketname` mit dem Befehl `apt show paketname` bekommt. Weitere Systeme zur Hilfe sind `man`, `help` und `info`.

Der Linux-Kernel und andere Teile des Betriebssystems, insbesondere Programme, aber auch Konfigurationsdateien und anderes, sowie viele Anwenderprogramme, werden durch *man pages* (Abkürzung von „manual pages“) kurz erläutert. Sie werden mit dem Befehl `man` aufgerufen. Für das Programm `wget` gibt zum Beispiel der Befehl

```
man wget
```

eine Erläuterung. Die Hilfstexte sind in komprimierten Dateien in Unterverzeichnissen des Verzeichnisses `/usr/share/man/` gespeichert und werden beim Aufruf des Befehls `man` von einem terminal pager angezeigt. Wenn kein anderer terminal pager bestimmt wird, ist dies `less` (den man durch Drücken der Taste `q` beendet).

Man pages werden in der Regel einer von acht sections zugeordnet. Jede section wird durch eine Zahl zwischen 1 und 8 gekennzeichnet (section number), welche in Klammern hinter dem Namen der man page steht, zum Beispiel `WGET(1)`, da `wget` seine man page in section 1 hat. Die section number ist aber für den Nutzer nicht wichtig.

Die man pages des Betriebssystems Debian (von dem sich Raspberry Pi OS ableitet) kann man auch im Internet auf der Webseite `https://manpages.debian.org/` lesen.

Man kann manpages auch in das PDF-Format konvertieren und speichern, obwohl möglicherweise einige Schriftarten nicht optimal wiedergegeben werden. Dies erreicht man zum Beispiel für das Programm `wget` mit

```
man -Tpdf wget > wget.pdf
```

wobei das Ergebnis in der Datei `wget.pdf` gespeichert wird.

Das Betriebssystem hat ein weiteres Dokumentationssystem, das weniger Themen behandelt als die man pages, aber dafür ausführlicher, nämlich `Info`. Es gehört zum GNU-Projekt. Das System arbeitet mit Hypertext, sodass man (ähnlich wie in der Wikipedia) über Verweise (links) zu anderen Textstellen springen kann. Es gibt mehrere Programme zum Lesen der Info-Seiten. In der Linux-Konsole kann man `info` (Standard) oder `pinfo` (im Stil des Browsers `lynx`) verwenden, die mit dem Befehl

```
sudo apt install info
```

beziehungsweise

```
sudo apt install pinfo
```

installiert werden. Information zum Programm `wget` erhält man zum Beispiel mit

```
pinfo wget
```

Durch Drücken der Taste `q` kann man `info` und `pinfo` beenden.

Für Befehle, die Teil der `bash` sind (built-in commands) erhält man mit dem built-in command `help` eine kurze Erklärung. Zum Beispiel gibt der Befehl

```
help alias
```

eine Erklärung zum built-in command `alias` der `bash`.

Programme stellen oft die Option `--help`, Kurzform `-h` bereit, die Hilfe ausgibt, insbesondere zu den Optionen, mit denen das Programm aufgerufen werden kann. Um einen langen Hilfstext im Terminal zu lesen, kann man die Ausgabe an einen terminal pager wie `less` weiterleiten, zum Beispiel für Hilfe zu den Optionen des Befehls `wget`:

```
wget -h | less
```

3.2 Terminals und der Linux-Framebuffer

3.2.1 Virtuelle Konsolen

Auf Seite 11 im Abschnitt 1.1.3 wurden die virtuelle Konsolen vorgestellt. Das sind Terminals, die in jedem Linux Betriebssystem, mit und ohne „Desktop“, zur Verfügung stehen. Im Betriebssystem Raspberry Pi OS gibt es normalerweise sechs virtuelle Konsolen. „Normalerweise“ heißt: Ihre Anzahl kann verändert werden. Die virtuellen Konsolen heißen `tty1`, `tty2`, `tty3`, `tty4`, `tty5` und `tty6`.

Mit der Tastenkombination `Ctrl Alt F2` (gleichzeitiges Drücken der Taste `Ctrl`, der Taste `Alt` und der Funktionstaste `F2`) gelangt man in die virtuelle Konsole `tty2`. Entsprechend geht es mit den anderen virtuellen Konsolen.

Im Betriebssystem Raspberry Pi OS *with desktop*, und auch in einigen anderen Linux Betriebssystemen mit „desktop“, gelangt man mit der Tastenkombination `Ctrl Alt F7` zum „desktop“. Bei manchen Linux Betriebssystemen mit „desktop“ muss man hierfür die Tastenkombination `Ctrl Alt F8` oder `Ctrl Alt F9` verwenden.

Da wir ohne X arbeiten, verwenden wir eine virtuelle Konsole. Sie wird im folgenden kurz Konsole oder Terminal genannt. Meistens sind wir in der Konsole `tty1`. Nur wenn eine zweite Konsole gebraucht wird, verwenden wir `tty2`. Linux ist (außer zu Beginn des Systemstarts) ein Mehrbenutzersystem (multi-user system). Daher können sich mehrere Nutzer (users) über jeweils eine Konsole anmelden, solange es noch freie Konsolen gibt, an denen kein anderer angemeldet ist.

Im Betriebssystem Raspberry Pi OS gelangt man mit der Tastenkombination `Alt` → (gleichzeitiges Drücken der Taste `Alt` und der Pfeiltaste →) zur nächsten virtuellen Konsole (zum Beispiel von `tty1` nach `tty2`). Mit der Tastenkombination `Alt` ← gelangt man entsprechend zur vorherigen virtuellen Konsole (zum Beispiel von `tty2` nach `tty1`)

Wenn, wie heute üblich, die Ausgabe einer Konsole durch einen Bildschirm dargestellt wird, arbeitet die Konsole im Framebuffer-Modus (siehe unten). Der zugehörige Terminal-Typ heißt `linux`; dieser Name ist in der Umgebungsvariablen `TERM` gespeichert (siehe Seite 127). Obwohl damit graphische Darstellungen und moderne Schriften (fonts) möglich sind, arbeitet die Konsole im Textmodus noch immer mit einem beschränkten Zeichensatz und wenigen Schriften. Dies behindert die Arbeit mit der Konsole erheblich und erfordert manchen Umweg, der bei Terminals im X-System wie zum Beispiel LXTerminal oder XTERM entfällt.

3.2.2 Direkte Ausgabe an den Framebuffer

Die Graphikausgabe eines Linux-Betriebssystems ohne X beruht auf einem Linux-Framebuffer, im Folgenden kurz Framebuffer genannt. Ein Framebuffer ist ein dateiähnliches Objekt, das als Schnittstelle zwischen Anwendungsprogrammen und einem Graphikausgabegerät dient, in der Regel dem Bildschirm.

In unserem Betriebssystem *Raspberry Pi OS Lite* sollte ein Framebuffer mit dem Pfad `/dev/fb0` bereits eingerichtet und verfügbar sein. Weitere Framebuffer sind möglich; alle haben einen Pfad `/dev/fb`*i*, wobei *i* eine natürliche Zahl ist. Wir verwenden im Folgenden jedoch nur `/dev/fb0` und nennen ihn kurz den Framebuffer. Außerdem gehen wir davon aus, dass der Bildschirm über den HDMI-Ausgang angeschlossen ist.

Normalerweise arbeiten wir im Terminal `tty1` (console 1) und der Befehl

```
tty
```

gibt `/dev/tty1` zurück. Mit dem Befehl

```
con2fbmap 1
```

erfahren wir, mit welchem Framebuffer `tty1` (console 1) zur Zeit arbeitet. Dies sollte `fb0` (framebuffer 0) sein.

Das Betriebssystem sorgt für eine eineindeutige Abbildung zwischen dem Arbeitsspeicherbereich, welcher den Bildschirminhalt (für den HDMI-Ausgang) steuert, und dem Framebuffer `/dev/fb0`. Durch Lesen des Framebuffers kann der Bildschirminhalt abgefragt werden und durch Schreiben des Framebuffers kann der Bildschirminhalt verändert werden. Der Framebuffer benötigt nicht das X-System. Graphische Programme ohne X, zum Beispiel `fbi` oder `fbterm` benutzen den Framebuffer. Man kann ihn auch mit der Programmbibliothek Pygame nutzen (siehe Seite 387).

Alle Farben werden durch Kombination von Rot (R), Grün (G) und Blau (B) erzeugt (RGB). Eine Anzeige, die jeweils 256 Werte (0 – 255) für R, G und B darstellen kann, heißt full-color system. Im framebuffer wird der Speicherplatz für eine Farbe als Kanal bezeichnet. In einem *true-color* System werden für jeden Bildpunkt (Pixel) alle Werte der drei Farbkanäle gespeichert. In einem true full-color system sind das 3 Byte pro Bildpunkt. Früher war der Speicherplatz kostbar und man hat versucht, die Farben mit weniger Bytes festzulegen. Dies geschah über color tables. Damit konnte man nur Farben verwenden, die in einer vorgegebenen Farbpalette enthalten waren.

Bei der Verwendung muss der Framebuffer an die Graphikausgabe (Bildschirm) angepasst sein. Dies geschieht durch Festlegung eines video mode, kurz Modus. In der Datei `/etc/fb.modes` ist eine Sammlung verschiedener benannter Modi.

Ein geeigneter Modus ist in der Regel nach dem Start des Betriebssystems bereits automatisch eingestellt worden.

In besonderen Fällen kann eine Änderung der automatischen Einstellung sinnvoll sein. Den aktuellen Modus und die Werte verschiedener Parameter erfährt man durch

```
fbset -i
```

Mit `fbset` kann man dem Framebuffer auch einen anderen benannten Modus zuweisen.

Die Ansteuerung des Framebuffers hängt vom Modus ab (siehe oben Seite 84). Im folgenden Beispiel gehen wir davon aus, dass der Modus "1680x1050" ist mit einer depth von 32 bpp (bits per pixel). In diesem Modus wird jeder Pixel durch vier Bytes dargestellt: je ein Byte pro Farbkanal (Blau, Grün und Rot) und ein Byte für den Alphakanal, der die Transparenz bestimmt. Mit einem Byte werden Werte zwischen $2^0 = 1$ und $2^8 = 255$ angegeben.

Eine Shell, wie die Bash, kann ein Byte innerhalb einer Zeichenkette in der Form `\x__` enthalten, wobei `__` aus zwei hexadezimalen Ziffern besteht. Eine Folge von vier Bytes kann in einer Zeichenkette als `\x__\x__\x__\x__` geschrieben werden. Die Farbe eines Pixels (Bildpunkts) ergibt sich aus den Werten der drei Farbkanäle, siehe Tabelle 3.2.

Farbe	hexadezimaler RGB Code	Zeichenkette für Framebuffer
blau	#0000FF	"\xFF\x00\x00\x00"
grün	#00FF00	"\x00\xFF\x00\x00"
gelb	#FFFF00	"\x00\xFF\xFF\x00"
rot	#FF0000	"\x00\x00\xFF\x00"
weiß	#FFFFFF	"\xFF\xFF\xFF\x00"

Tabelle 3.2: Zeichenketten für Pixel im Framebuffer, Modus "1680x1050"

Folgendes Script gibt den vierfarbigen Umriss eines Quadrats auf dem Bildschirm aus.

```
#!/usr/bin/env bash
# Example for framebuffer usage (AHG, 2021)
# framebuffer parameters: width (pixels), bytes per pixel
fb_width=1680 ; fb_bytespp=4
# green (#00FF00), H yellow (#00FFFF), red (#FF0000), white (#FFFFFF)
c_g="\x00\xFF\x00\x00" ; c_y="\x00\xFF\xFF\x00"
c_r="\x00\x00\xFF\x00" ; c_w="\xFF\xFF\xFF\x00"
# define function fbw, which writes to the framebuffer
# parameters: $1: x-value, $2: y-value, $3: color
fbw(){
printf "$3" | dd bs=$fb_bytespp \
seek=$(($2*$fb_width+$1)) } of=/dev/fb0 &>/dev/null
}
# plot colored line by calling function fbw
typeset -i n x1 y1 x2 y2 ; clear
x1 = 100 ; y1 = 100 ; x2 = 200 ; y2 = 200
n=$x1 ; while [ $n -le $x2 ] ; do fbw $n $y1 $c_g ; n=$n+1 ; done
n=$y1 ; while [ $n -le $y2 ] ; do fbw $x2 $n $c_y ; n=$n+1 ; done
n=$x2 ; while [ $n -ge $x1 ] ; do fbw $n $y2 $c_r ; n=$n-1 ; done
n=$y2 ; while [ $n -ge $y1 ] ; do fbw $x1 $n $c_w ; n=$n-1 ; done
```

Der dd-Befehl kopiert Daten in den Framebuffer, denn `of=/dev/fb0`. Die Eingabe (input) des dd-Befehl wird über eine Pipe `|` von einer Zeichenkette geliefert, die 4 Bytes enthält. Weiterhin ist `bs` (block size) die Anzahl der Bytes, die gelesen und geschrieben werden (Bytes pro Bildpunkt), und `seek` ist die Anzahl der bs-Blöcke, die vor dem Schreiben übersprungen werden (Position im Framebuffer).

Zum Vergleich nehmen wir nun einen Framebuffer im Modus "1920x1080" mit einer depth von 16 bpp (bits per pixel).

```
#!/usr/bin/env bash
fb_width=1920 ; fb_bytespp=2
c_red="\x00\xF8"     # rot
c_orange="\x20\xFD"  # orange
c_yellow="\xE0\xFF"  # gelb
c_green="\xE0\x07"   # grün
c_cyan="\xFF\x07"    # cyan
c_blue="\x1F\x00"    # blau
c_white="\xFF\xFF"   # weiß
c_cyan="\xFF\x07"    # cyan
# define function fbw, which writes to the framebuffer
# parameters: $1: x-value, $2: y-value, $3: color
fbw(){
printf "$3" | dd bs=$fb_bytespp \
seek=$(($2*$fb_width+$1)) of=/dev/fb0 &>/dev/null
}
# plot colored line by calling function fbw
typeset -i n x1 y1 x2 y2 ; clear
x1=100 ; y1=100 ; x2=200 ; y2=200
#
n=$x1 ; while [ $n -le $x2 ] ; do fbw $n $y1 $c_white ; n=$n+1 ; done
n=$y1 ; while [ $n -le $y2 ] ; do fbw $x2 $n $c_orange ; n=$n+1 ; done
n=$x2 ; while [ $n -ge $x1 ] ; do fbw $n $y2 $c_blue ; n=$n-1 ; done
n=$y2 ; while [ $n -ge $y1 ] ; do fbw $x1 $n $c_red ; n=$n-1 ; done
```

Zu beachten ist, dass die Byte-Reihenfolge (endianness) der Farben umgekehrt sein kann, zum Beispiel `"\x00\xF8"` statt `"\xF8\x00"` für Rot.

3.2.3 Graphisches Terminal fbterm

Die Konsole bietet nur wenige Schriften (console fonts) zur Textausgabe. Man kann aber das Terminalprogramm `fbterm` installieren, das mit dem Framebuffer arbeitet und skalierbare Schriften bietet (GUI fonts). Es ist insofern mit den Terminals vergleichbar, die unter X oder Wayland laufen. Das Programm `ucimf` dient der Eingabe von Unicode-Zeichen. Um es in `fbterm` nutzen zu können, wird auch `fbterm-ucimf` installiert.

```
sudo apt install fbterm fbterm-ucimf
```

Das Paket fontconfig liefert Hilfsprogramme, um die verfügbaren Schriften anzuzeigen.

```
sudo apt install fontconfig
```

Nun erhalten wir eine Liste der Schriften (GUI fonts) mit dem Befehl

```
fc-list | less
```

Nach seiner Installation wird `fbterm` (ohne `ucimf`) mit dem Befehl

```
fbterm
```

aufgerufen. Zum Verlassen von `fbterm` und Rückkehr zur Konsole dient der Befehl

```
exit
```

Wenn der Framebuffer in einem hochauflösenden Modus arbeitet, erscheint die Schrift von `fbterm` in der Standardeinstellung zu klein. Es ist dann sinnvoll, die Konfigurationsdatei `.fbtermrc` zu ändern, zum Beispiel mit dem Editor `nano`.

Man beachte, dass Kommentare, die mit dem Rautezeichen `#` beginnen, nur am Anfang einer Zeile erlaubt sind (die ganze Zeile ist Kommentar), aber nicht nach einer Zuweisung.

Die beiden Zeilen, die mit `font-names=` und `font-size` beginnen, kann man in

```
font-names=DejaVuSansMono,mono
font-size=28
```

ändern. Man kann auch einen anderen Mono-Font als erste Wahl benennen oder eine andere Größe (size) wählen.

Will man eine andere Standard-Schriftfarbe, zum Beispiel grün, ändert man die Zeile, die mit `color-foreground=` beginnt, in

```
color-foreground=2
```

Man erkennt dann schon an der Farbe, ob man in einer Konsole oder in `fbterm` arbeitet. Wem das Grün zu dunkel ist, kann im `fbterm` die Farbpalette mit dem Befehl

```
echo -en "\e]P290FF90"
```

ändern und Grün in Hellgrün (light green) verwandeln.

Wenn das Paket `gpm` installiert wurde, und eine Maus angeschlossen ist und konfiguriert wurde, kann man, wie auf Seite 48 beschrieben, copy & paste benutzen. Man markiert eine Zeichenkette auf dem Bildschirm mit der Maus, während man die linke Maustaste gedrückt hält. Im Unterschied zur Konsole geschieht das Einfügen der markierten Zeichenkette durch Drücken der rechten Maustaste.

Tastenkombinationen (keyboard shortcuts) zur Steuerung von `fbterm` werden mit

```
sudo chmod u+s /usr/bin/fbterm
```

ermöglicht. Wer die damit verbundene Einschränkung der Systemsicherheit scheut, kann auf die Tastenkombinationen verzichten. Ohne Freigabe der Setzung von Tastenkombinationen zeigt `fbterm` nach dem Start einen entsprechenden Hinweis. Die verfügbaren Tastenkombinationen und weitere Information erhält man mit

```
man fbterm
```

Um `fbterm` mit `ucimf` zu nutzen, müssen Tastenkombinationen (keyboard shortcuts) möglich sein (siehe oben). In der Konfigurationsdatei `.fbtermrc` wird die Zeile, die mit `input-method=` beginnt, in

```
input-method=fbterm_ucimf
```

geändert. Die Grundeinstellungen (setup) für `ucimf` werden mit

```
ucimf_start
```

in einem Menü vorgenommen. Die Tastenkombination `Ctrl-SPACE`, also gleichzeitiges Drücken der Taste `Ctrl` (`Strg`) und der Leertaste, startet und beendet in `fbterm` die Eingabemethode. Nach dem Beenden bleiben zwei seltsame Zeichen stehen, aber durch Drücken der ENTER-Taste erhalten wir wieder eine normale Kommandozeile.

3.3 Dateisysteme und der Verzeichnisbaum

3.3.1 Dateisysteme und das Einhängen (Mounten)

Gerätedateien

Datenträger sind Peripheriegeräte (peripherals) zur Speicherung von Daten außerhalb der CPU. Der Zugriff auf ein Peripheriegerät erfolgt über eine *Gerätedatei* (device file), die sich gegenüber dem Nutzer wie eine normale Datei verhält. Gerätedateien liegen meistens im Verzeichnis `/dev`. Beispiel: Der Pfad zur Gerätdatei einer Speicherkarte im Raspberry Pi könnte zum Beispiel `/dev/mmcblk0` sein.

Die Gerätedatei eines Datenträgers ist blockorientiert. Blockorientierte Gerätedateien (block special files) heißen *block devices*. In block devices erfolgt das Lesen und Schreiben von Datenblöcken über einen Puffer (Zwischenspeicher), der vom Betriebssystem verwaltet wird, was der Nutzer meistens nicht bemerkt. Die Größe eines Datenblocks (block size) beträgt oft 512 Byte oder 4096 Byte; in Audio-CDs üblicherweise 2352 Byte.

Grundsätzlich kann man auf ein block device direkt lesen und schreiben, zum Beispiel mit dem Dienstprogramm `dd`, ohne die unten beschriebene Partitionierung und Formatierung. Das ist aber nur praktikabel, wenn ein einziger, zusammenhängender Datensatz geschrieben und gelesen wird.

Partitionen

In der Regel wird ein block device (die Gerätedatei eines Datenträgers) in nicht überlappende Abschnitte unterteilt, welche *Partitionen* genannt werden. Jeder Partition entspricht auf dem Datenträger ein zusammenhängender physikalischer Speicherbereich. Der Nutzer interessiert sich meistens nicht für den physikalischen Ort und die Art der Datenspeicherung, sondern will die Daten gedanklich oder logisch gliedern. Das Betriebssystem stellt dafür sogenannte *Dateisysteme* zur Verfügung.

Jede Partition erhält ein Dateisystem zur Verwaltung der Dateien, die in der Partition gespeichert sind. Es ist grundsätzlich möglich, auf die Partitionierung zu verzichten und ein Dateisystem direkt auf einem block device einzurichten, aber das ist unüblich und führt daher zu Schwierigkeiten. Es ist jedoch möglich und oft sinnvoll, nur eine Partition für das ganze block device einzurichten.

Dateisysteme

In einem block device hat in der Regel jede Partition ein Dateisystem. Es gibt verschiedene Dateisysteme, zum Beispiel `ext3`, `ext4`, `FAT32` und `swap`. Dem logischen Dateisystem ist physikalischer Speicherplatz auf dem Datenträger zugeordnet, ohne dass der Nutzer dies beachten muss: Die Verwaltung übernimmt das Betriebssystem.

Wenn der Rechner nicht vorschriftsmäßig heruntergefahren wird (zum Beispiel durch `poweroff`), sondern durch einen Stromausfall ausgeschaltet wird, kann beim nächsten Hochfahren die Fehlermeldung „No init found.“ erscheinen und der Startvorgang hält an. Oft kann das System gerettet werden, indem man das Dateisystem in einem anderen Linux-Rechner prüfen lässt. *Bevor* wir die Speicherkarte in das Lesegerät des anderen Linuxrechner einstecken, geben wir in einem Terminal des anderen Rechners den Befehl

```
lsblk -p
```

ein, um die Speichermedien des Rechners aufzulisten. Dann stecken wir die Speicherkarte in das Lesegerät und führen obigen Befehl erneut aus. Das zusätzlich angezeigte Gerät ist die Speicherkarte und könnte zum Beispiel `sda` heißen, muss aber nicht. Wir kürzen den Namen hier mit `X` ab und notieren ihn. In der rechten Spalte der Auflistung steht der „mountpoint“ für die eingehängten Partitionen. Falls Partitionen der Speicherkarte eingehängt sind, werden sie der Reihe nach mit dem Befehl `umount Y` ausgehängt, wobei `Y` der Name der Partition ist. Nun geben wir den Befehl

```
sudo fdisk -l /dev/X
```

ein, wobei `l` der Kleinbuchstabe l und `X` der Name der Speicherkarte sind. Danach wird die Speicherkarte aus dem Lesegerät des anderen Linux-Rechners genommen und wieder in den Raspberry eingesteckt. Nun können wir einen neuen Systemstart versuchen.

Manuelles Mounten eines USB-Sticks

Wir schließen zum Beispiel einen USB-Stick an einen freien USB-Port an. Der USB-Stick enthält eine oder mehrere Partitionen und jede Partition hat ein Dateisystem. Bei USB-Sticks ist es es oft eine Partition mit dem Dateisystem `vfat`. Der Befehl

```
lsblk -f
```

zeigt die Gerätenamen der angeschlossenen Datenträger (Festplatten, USB-Sticks, SD-Karten), deren Partitionen und die Dateisysteme.

Wir wollen das Dateisystem einer Partition auf dem USB-Stick in das allgemeine Dateisystem unseres Betriebssystems „einhängen“, so dass man die Dateien lesen und schreiben kann. Dieses Einhängen heißt (schlecht eingedeutscht) auch *mounten* und erfolgt bei Raspbian Lite (im Gegensatz zu Raspbian mit Desktop) zunächst nicht automatisch. Man kann ein automatisches Mounten einrichten oder das Mounten eines Dateisystems „manuell“, also Schritt für Schritt, selbst vornehmen.

Jeder Partition auf dem externen Datenträger wird ein neues Verzeichnis im allgemeinen Dateisystem zugeordnet. Dafür erzeugen wir ein Verzeichnis mit einem beliebigen Namen (hier: `usb_1`). Es ist üblich. ein Unterverzeichnis von `/media` zu erzeugen.

```
sudo mkdir /media/usb_1
```

Wenn der Datenträger an einem USB-Port angeschlossen wird, sollte das System ihn als Gerät erkennen, einen Namen (zum Beispiel `sda` zuweisen und, wenn das Dateisystem auf dem Datenträger erkannt wird auch die Partitionen benennen (zum Beispiel `sda1`). Es gibt verschiedene Möglichkeiten, diese Vorgänge sichtbar zu machen. Am einfachsten geht es mit dem oben genannten Befehl `lsblk -f` . Eine Alternative ist

```
sudo fdisk -l | grep sd
```

Dabei ist `|` ein Sonderzeichen, das auf den meisten Tastaturen durch die Tastenkombination `AltGr <` erzeugt wird. Durch Anhängen von `| grep sd` erreichen wir, dass nicht die ganze lange Ausgabe des Befehls `fdisk -l` gezeigt wird, sondern nur Zeilen, die das das Buchstabenpaar `sd` enthalten. Vor Anschluss des externen Datenträgers gibt der Befehl nichts aus, aber nach Anschluss des Datenträgers zeigt er einen Gerätenamen für den Datenträger (hier `sda`) und die darauf befindlichen Partitionen (hier `sda1`).

Nun können wir den Datenträger, genauer: seine Partition `sda1`, mounten.

```
sudo mount -o uid=pi,gid=pi /dev/sda1 /media/usb_1
```

Das Dateisystem (hier `vfat`) der Partition (hier: `sda1`) wird in der Regel automatisch erkannt. Das beim Mounten angegebene Verzeichnis (hier `/media/usb_1`) heißt Mountpoint oder Einhängepunkt. Ob wir über den Berg sind, kann man mit

```
sudo mount | grep sda1
```

prüfen. Nun ist die Partition `sda1` des Datenträger `sda` als Verzeichnis `/media/usb_1` Teil des allgemeinen Dateisystems geworden.

Wenn man den oben angegebenen langen Befehl zum Mounten öfter braucht, sollte man einen Kurzbefehl dafür einrichten (siehe Seite 51).

Das Dateisystem `vfat` kennt kein Rechtesystem, wie es bei Linux-Dateisystemen üblich ist. Durch die Angabe `-o uid=pi,gid=pi` legen wir fest, dass der Nutzer `pi` auf die Partition `sda1` des Datenträgers zugreifen, und Dateien lesen und schreiben darf. Außerdem kann das natürlich auch der Administrator.

Information über das Dateisystem und den Speicher aller USB-Datenträger, die im Verzeichnis `media` „gemounted“ wurden (siehe oben), kann mit dem Befehl

```
df -Th | grep media
```

auf dem Bildschirm ausgegeben werden. In der dritten Spalte steht die gesamte Speichergröße, in der vierten der benutzte und in der fünften Spalte der verfügbare Speicherplatz.

Wichtig ist, dass der Datenträger aus dem Dateisystem ausgehängt wird, *unmounten* genannt, *bevor* er vom vom USB-Port entfernt wird! Unmounten ist aber nur möglich, wenn kein Zugriff mehr auf das Datenträgerverzeichnis und seine Dateien mehr erfolgt. Insbesondere darf das Datenträgerverzeichnis dann nicht Arbeitsverzeichnis sein.

```
sudo umount /media/usb_1
```

Der Befehl für das Unmounten ist also `umount`. Auch für das Unmounten kann an einen Kurzbefehl definieren (siehe Seite 51).

Achtung: Entfernt man den Datenträger ohne Unmounten, kann das Dateisystem beschädigt werden!

3.3.2 Verzeichnisbaum und Dateien

Unterverzeichnisse von /

In einem Linuxsystem, mit oder ohne Desktop, wird das Hauptverzeichnis / meistens in Unterverzeichnissen gegliedert:

`/bin`	Binärdateien, insbesondere für Befehle
`/boot`	Systemstartdateien und der Kernel
`/dev`	Gerätedateien = device files (Treiber)
`/etc`	Info- und Konfigurationsdateien
`/home`	Verzeichnisse der einzelnen Nutzer
`/lib`	Bibliotheken, die bei Bedarf geladen werden
`/lost+found`	Dateireste, die nach Systemabsturz bleiben
`/media`	eingehängte (gemountete) Wechseldatenträger
`/mnt`	eingehängte Datenträger, wenn nicht in `/media`
`/opt`	zusätzliche Dateien für das Betriebssystem

`/proc`	Info zu Betriebssystem und Prozessen
`/root`	Verzeichnis für den Administrator root
`/run`	während der Laufzeit von Programmen genutzt
`/sbin`	Dienstprogramme zur Systemverwaltung
`/srv`	Dateien von lokalen Servern (falls es sie gibt)
`/tmp`	vorübergehende Ablage von Dateien
`/usr`	installierte Software (mit `/usr/bin` und `/usr/lib`)
`/var`	Konfigurationsdateien und Systemeinstellungen

Dateinamen

Dateien wurden bereits früher definiert (siehe Seite 7) und wir ergänzen hier etwas über Dateinamen, die vom Nutzer vergeben werden. Sie sollten so gewählt werden, dass sie nicht nur in Linux, sondern auch in anderen Betriebssystemen gültig sind und nur Zeichen enthalten, die sowohl auf einer deutschen als auch auf einer englischen Tastatur verfügbar sind.

Zu empfehlen ist die Beschränkung auf alphanumerische Zeichen (Buchstaben und Ziffern), Bindestrich - und Unterstrich _ . Da Kleinweich-Betriebssysteme (anders als Linux) bei Dateinamen nicht zwischen zwischen Groß- und Kleinschreibung unterscheiden, sollte man nur Kleinbuchstaben verwenden.

Dateinamen sollten nicht zu lang werden und doch einen erkennbaren Hinweis auf den Inhalt geben. Oft ist es sinnvoll, wenn der Nutzername abgekürzt am Anfang des Dateinamens steht und ein 6-stelliges Datum in der Form YYMMDD am Ende, wobei YY die zweistellige Jahreszahl, MM die zweistellige Monatszahl und DD die zweistellige Tageszahl sind. Die Reihenfolge YYMMDD führt zu eine sinnvollen Sortierung von Dateien, deren Namen sich nur im Datumsteil unterscheiden. Manchmal, zum Beispiel bei Serien von Photodateien, hängt man entsprechend noch die Uhrzeit an.

Statt Dateinamen kann man oft ein Dateinamenmuster angeben, das auf eine oder mehrere Dateien passt. Das Muster kann unter anderem folgende „Platzhalter“ enthalten:

`*`	passt auf jede Zeichenkette
`?`	passt auf ein beliebiges Zeichen
`[ ... ]`	passt auf jedes Zeichen innerhalb der Klammern
~	passt auf das Heimatverzeichnis des Nutzers

Da die genannten Zeichen eine besondere Bedeutung in Mustern haben, sollten sie nicht in einem Dateinamen verwendet werden.

3.3.3 GParted oder parted und mkfs

Die Programme `GParted` und `parted` ermöglichen die Einrichtung und Verwaltung von Partitionen von Datenträgern (wie Festplatten und USB-Sticks). `Gparted` arbeitet nur in Linux-Betriebssystemen mit graphischer Oberfläche, `parted` in der Kommandozeile. Bei beiden Programmen besteht die Gefahr, dass bei Fehlern alle Daten unbrauchbar

werden und darum dürfen sie nur mit Administratorrechten benutzt werden. Nach der Partitionierung des Datenträgers bekommt jede Partition ein Dateisystem. Dies nennt man *Formatierung*. Während `GParted` Partitionierung und Formatierung übernimmt, muss nach `parted` noch ein zweites Programm zur Formatierung, zum Beispiel eine Variante von `mkfs`, aufgerufen werden, auch mit Administratorrechten.

Die Verwendung von `GParted` ist einfacher. Wer ein Linux-Betriebssystemen mit graphischer Oberfläche benutzen kann, sollte besser `GParted` statt `parted` anwenden. Wir zeigen die Benutzung der Programme nur an Beispielen und verweisen ansonsten auf die `Dokumentation für GParted` beziehungsweise die Dokumentation für `parted`.

Festplattenformatierung mit GParted

Eine externe Festplatte, die unter Linux benutzt wird, sollte ein Linux-Dateisystem tragen. Das gilt natürlich auch für den Raspberry Pi. Die Erstellung eines Dateisystems nennt man Formatieren. Gleiches gilt für andere Datenträger, aber wir sprechen im Folgenden über Festplatten.

Auf Linux-Betriebssystemen mit graphischer Oberfläche kann man dafür das Programm *GParted* benutzen. Es muss mit Administratorrechten aufgerufen werden. Da beim Formatieren alle Daten der Festplatte verloren gehen, ist es notwendig, unbedingt die richtige Festplatte zur Formatierung auszuwählen. Dann wählt man im Reiter `Laufwerk` die Option `Partitionstabelle erstellen ...` und `msdos` (dies erzeugt einen sogenannten MBR nach Art des eines früher verbreiteten Betriebssystems DOS; alternativ könnte man `gpt` verwenden) und klickt auf `Anwenden`. Nun wählt man `Neu` im Reiter `Partition`. Es öffnet sich ein neues Fenster mit Voreinstellungen, welche man übernehmen kann, wenn man nur eine Partition auf der Festplatte haben will. Als Dateisystem kann man `ext4` nehmen und es ist nützlich, im Feld nach `Bezeichnung:` einen Namen zu vergeben. Man sollte nur Kleinbuchstaben verwenden, ohne Umlaute und sonstige Sonderzeichen; auch ein Unterstrich _ kann enthalten sein. Wenn man sicher ist, alle Eintragungen richtig vorgenommen zu haben, wählt man `Alle Operationen ausführen` im Reiter `Bearbeiten` und klickt aus `Anwenden`. Danach kann man auf `Schließen` klicken und dann *GParted* beenden.

Beim Erstellen der Partitionen mit dem Dateisystem `ext4` bekommt jede Partition ein Verzeichnis `lost+found`, in dem Dateien abgelegt werden, die – soweit möglich – automatisch (durch das Programm `fsck`) nach einer möglichen Beschädigung wiederhergestellt wurden, zum Beispiel nach einem Systemabsturz.

Nach dem Formatieren mit *GParted* gehören die Daten auf der Festplatte dem Administrator `root` und ein anderer user darf nicht darauf zugreifen. Um nun alle Daten für alle vollständig frei zu geben, geht man in in einem Terminal in das Systemverzeichnis, in welchem die Festplatte gemounted ist (DEBIAN und UBUNTU: `/media`, SUSE: `/run/media`) und gibt (mit Admistratorrechten) den Befehl

```
sudo chmod 777 FP
```

ein, wobei `FP` die Bezeichnung der Festplatte ist, die mit *GParted* festgelegt wurde. Warnung: Nun darf jeder auf der Festplatte lesen, schreiben und ausführen. Wenn die

Festplatte nicht dem Administrator, sondern einem user gehören soll, ändert man dies mit dem Befehl `chown` (den wir an dieser Stelle nicht beschreiben).

Formatierung eines USB-Sticks mit parted

In einem Beispiel werden Partitionierung und Formatierung eines USB-Sticks zur Datenspeicherung (nicht Booten eines Betriebssystems) in unserem Raspberry Pi mit `parted` und `mkfs` gezeigt. Wir schließen den USB-Stick noch nicht an und geben den Befehl

```
lsblk -n -d
```

der eine Liste aller direkt (nicht über ein Netzwerk) angeschlossenen, blockorientierten Geräte ausgibt. Die Option `-n` unterdrückt die Ausgabe einer Kopfzeile und die Option `-d` unterdrückt die Ausgabe der Partitionen. Beide Optionen kann man auch weglassen. Dann wird der USB-Stick angeschlossen und der Befehl

```
lsblk -n -d
```

wiederholt. Das neu hinzugekommene Gerät ist der USB-Stick, in diesem Beispiel `sdb`, aber natürlich könnte es auch ein anderer Name sein. Warnung: Die folgenden Befehle partitionieren und formatieren den USB-Stick und alle darauf enthaltenen Daten können unbrauchbar werden! Eine zuverlässige Löschung vertraulicher Daten wird dadurch aber nicht bewirkt. Der Befehl

```
sudo parted /dev/sdb mklabel msdos
```

erstellt eine Partitionstabelle vom Typ `msdos`. Die Warnung „Warning ... continue?“ beantworten wir mit `y` und Drücken die ENTER Taste. Die Meldung „Information: ... fstab.“ interessiert uns hier nicht. Der Befehl

```
sudo parted /dev/sdb mkpart primary fat32 2048 100%
```

erzeugt eine Partition `sdb1` und bereitet die Formatierung mit dem Dateisystem `fat32` vor, führt diese aber noch nicht aus. Die Wahl des besten Startwerts, in diesem Beispiel `2048`, ist schwierig und wird hier nicht erörtert. Die Meldung „Information: ... fstab.“ interessiert uns hier nicht. Der Befehl

```
sudo mkfs.vfat -F32 /dev/sdb1
```

erzeugt das Dateisystem `fat32` in der eben erzeugten Partition `sdb1`. Mit dem Befehl

```
sudo parted /dev/sdb print
```

geben wir eine Übersicht über die erfolgte Partitionierung und Formatierung aus. Die Vorbereitungen für das manuelle Mounten, sowie das Mounten (Einhängen) und Unmounten (Aushängen) wurden oben bereits beschrieben, sieh Seite 90.

3.4 Systemnahe Programmiersprache C

C, C++ und C# sind drei verschiedene Programmiersprachen. Wir behandeln hier nur C, eine Low-Level-Programmiersprache. Das heißt: sie arbeitet nah an einfachen Prozessorbefehlen und ohne Objektorientierung. C Programme sind weniger anschaulich als zum Beispiel Pythonprogramme, können aber sehr schnell ausgeführt werden.

3.4.1 Einführung in C

Die Programmiersprache C wurde Anfang der 1970er Jahren entwickelt und wird für Betriebssysteme, einschließlich Linux, und systemnahe, schnell laufende Anwendungen auf allen Rechnertypen eingesetzt, von Embedded systems bis Supercomputern.

Vom Quellcode zum ausführbaren Programm

Ein Computerprogramm in der Sprache C wird zunächst in Quellcode für Menschen lesbar geschrieben und in einer oder mehreren Dateien dauerhaft gespeichert. Die Endung `.c` ist üblich für Quellcodedateien in C (und `.cpp` für Quellcodedateien in C++).

Ein C Programm besteht aus Funktionen und *externen Variablen* (Variablen, die außerhalb von Funktionen definiert werden). Letztere wirken *global*, das heißt: sie sind im ganzen Programm gültig. Innerhalb von Funktionen definierte (interne) Variablen wirken dagegen *lokal*, das heißt: sie sind nur innerhalb der betreffenden Funktion gültig. Eine *Funktion* fasst einen mehrfach verwendbaren Programmteil unter einem Namen zusammen und wird mit ihrem Namen aufgerufen, wobei variable Werte (Parameter) übergeben werden können. Sie kann ein Ergebnis zurückgeben (Rückgabewert). Nach der Ausführung der Funktion wird das Programm mit dem Befehl fortgesetzt. der auf den Funktionsaufruf folgt. In einer Funktion können Funktionen aufgerufen werden, aber nicht definiert werden (Funktionsdefinitionen sind extern).[1] Jedes C Programm muss genau eine Funktion namens *main* enthalten. Sie wird zu Beginn der Programmausführung automatisch aufgerufen.

Aus dem Quellcode macht ein *Compiler* genanntes, mehrteiliges Übersetzungsprogramm *Objektcode*, das heißt: Maschinencode für einen bestimmten Rechner. In unserem Betriebssystem wird der Compiler `gcc` verwendet. Der Teil davon, welcher zuerst ausgeführt wird, heißt *Präprozessor*. In der Regel wird zunächst für jede Quellcodedatei eine Objektcodedatei erzeugt. Die Endungen `.o` und `.obj` sind üblich für Objektcodedateien.

Der Objektcode wird mit dem Laufzeitsystem des Compilers, welches zusätzlichen Maschinencode für den Programmablauf bereitstellt, durch den *Linker* zusammengebunden. Der Linker ist Teil des Compilers. So entsteht ein ausführbares (englisch: executable) Programm , das vom gcc Compiler standardmäßig in einer Datei namens `a.out` gespeichert wird. Meistens wählt man aber einen anderen Namen für die ausführbare Datei, wobei in UNIX und Linux keine Dateinamenendung üblich ist, während andere Betriebssysteme zum Beispiel die Endung `.exe` empfehlen.

[1] In einem C Programm kann auch die Funktion `main` innerhalb der Funktion `main` aufgerufen werden.

Kurzbeschreibung und C Standard-Bibliothek

In der Regel muss man in den Quellcode Dateien einbinden, welche besondere Funktionen bereitstellen (Header-Dateien). Einige dieser Dateien sind bereits Teil der C Standard-Bibliothek, siehe Tabelle 3.3.

locale.h	Ortsübliche Einstellungen: Sprache, Region und Kodierung
math.h	mathematische Funktionen und Konstanten
stdbool.h	Datentyp `bool` für Wahrheitswerte (true, false)
stdio.h	Eingabe und Ausgabe (Tastatur, Bildschirm, Datei)
string.h	Verarbeitung von Zeichenketten
time.h	Datum und Uhrzeit
utsname.h	Information zu Betriebssystem und Rechnerarchitektur

Tabelle 3.3: Teilmenge der C Standard-Bibliothek

Einige Funktionen aus der C Standard-Bibliothek sind in Tabelle 3.4 aufgeführt.

fgets	`char txt[81]; fgets(txt, 81, stdin);`
M_PI	`printf("Pi ist etwa %f.\n", M_PI);`
M_SQRT2	`printf("Die positive Wurzel aus 2 ist %f.\n", M_SQRT2);`
printf	`int i = 2; printf("Ver%dfelt nicht!\n", i);`
time	`time_t sek; sek = time(NULL);`

Tabelle 3.4: Beispiele von Konstanten und Funktionen aus der C Standard-Bibliothek

Die Funktion `printf` aus `stdio.h` gibt im Beispiel der Tabelle 3.4 einen Text auf dem Bildschirm aus, in welchem `%d` durch eine Ganzzahl ersetzt wird. Entsprechend würde `%f` durch eine Gleitkommazahl, `%c` durch ein Zeichen und `%s` durch eine Zeichenkette ersetzt. `%ld` steht für eine große Ganzzahl (long int). Rückgabewert von `printf` ist die Anzahl der ausgegebenen Zeichen, oder eine negative Zahl im Fehlerfall.

Die Funktion `time` aus `time.h` gibt mit dem Argument `NULL` die Anzahl der Sekunden zurück, die seit dem 1. Januar 1970 vergangen sind (siehe Unix time, Seite 298). Der Rückgabewert hat den Datentyp time_t, welcher vom gcc Compiler als ganzzahliger Datentyp mit Vorzeichen definiert wird, sodass negative Werte möglich sind.

```
#include <stdio.h>
#include <time.h>
int main () {
  time_t sek = time(NULL);
  printf("Seit 01.01.1970 sind %ld Sekunden vergangen.\n", sek);
  return 0;
}
```

Zur Compilierung von obigem Programm, das in der Datei `zeit.c` gespeichert wurde, in ein ausführbares Programm namens `zeit`, gibt man den Befehl

```
gcc -o zeit zeit.c
```

ein. Zur Ausführung des Programms genügt dann der Befehl

```
./zeit
```

Zeitangaben im Stil des Landes, für das unser Betriebssystem konfiguriert wurde (Deutschland), erhält man mit Hilfe der Dateien `locale.h` und `time.h`. Die Funktion `strftime` aus `time.h` bildet einen Ausgabestring aus einer Zeitangabe, einem Formatstring (Zeichenkette, welche hier ein Zeitformat beschreibt) und einer Zeichenkette für die Locale. Mit der Locale `de_DE.UTF-8` steht im Formatstring `%A` für den Wochentagsnamen, `%x` für das Datum `DD.MM.YYYY` (mit Tag `DD`, Monat `MM` und Jahr `YYYY`), und `%X` für die Zeit `HH.MM.SS` (mit Stunde `HH`, Minute `MM` und Sekunde `SS`). Ein Beispiel:

```
#include <locale.h>
#include <stdio.h>
#include <time.h>
int main() {
  char *locale;
  locale = setlocale(LC_ALL, "");
  printf("Locale (Sprache_Region.Kodierung): %s\n", locale);

  char zeit[256]; time_t t = time(NULL);
  strftime(zeit, sizeof(zeit), "%A, %x, %X", localtime(&t));
  printf("Heute ist %s.\n", zeit);
  return 0;
}
```

Das folgende Beispiel zeigt, wie man eine Texteingabe vom Nutzer abfragt (über die Tastatur, daher `stdin` in der sechsten Programmzeile), welche auch Leerzeichen enthalten darf. Es sind nur einzeilige Texteingaben möglich, denn der Lesevorgang endet bei einem Newline-Zeichen (Zeilenumbruch) oder einem EOF-Zeichen.

```
#include <stdio.h>
#include <string.h>
int main() {
  char txt[10];   // Nullzeichen 0 (NULL) ist nicht die Ziffer 0
  printf("Schreib was (max. 8 Zeichen): ");
  fgets(txt, 10, stdin); // max. 8 Zeichen + Zeilenumbruch + NULL
  if ( strchr(txt, '\n') != 0 ) { txt[strcspn(txt,"\n")] = 0; }
  else { txt[8] = 0; }   // das 9. Zeichen wird zum Nullzeichen
  printf("Eingegeben (erste 8 Zeichen): %s\n", txt);
  return 0 ;
}
```

Die Funktion `fgets` wird in der Header-Datei `stdio.h` definiert. Wichtig ist, das `char` array genügend groß zu wählen, damit der Text vollständig aufgenommen werden kann. Da maximal 8 Zeichen plus ein Zeilenumbruch eingegeben werden dürfen und diese Zeichenkette automatisch mit einem Nullzeichen \0 abgeschlossen wird, ist die Länge des `char` array um 2 größer als die maximale Länge der Texteingabe, also 10. Falls 9 Zeichen plus Zeilenumbruch eingeben werden, wird der Zeilenumbruch nicht mehr gespeichert. Dann wir das neunte Zeichen (mit dem Index 8) durch ein Nullzeichen ersetzt.

Im folgenden Beispiel wird ein Text von der Tastatur eingelesen und in der Variablen `txt` gespeichert, welcher Leerzeichen und Tabulatorzeichen enthalten kann. Dann werden mit der Funktion `rems` alle Leerzeichen und Tabulatorzeichen entfernt und der veränderte Text auf dem Monitor ausgegeben.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_TXT 82 // 80 + newline + NULL
void rems(char * str); // declaration of function

int main() {
// settings
char txt[MAX_TXT];
// read max. MAX_TXT - 2 characters
printf("Old text: "); fgets(txt,MAX_TXT,stdin);
if ( strchr(txt, '\n') != NULL )
   { txt[strcspn(txt,"\n")] = 0; }
else
   { printf ("ERROR: STRING TOO LONG!\n\n");
     exit(EXIT_FAILURE); }
// write text without spaces and tabs
rems(txt);
printf("New text: %s\n",txt);
} // end of main

// Function to remove all spaces and tabs
void rems(char* s) {
char* d = s;
do {
   while ( *d == ' ' || *d == '\t' )
   { ++d; }
   }
while (*s++ = *d++);
} // end of rems
```

Die Header-Datei `utsname.h` liefert Information zum Betriebssystem. Beispiel:

```
#include<stdio.h>
#include<sys/utsname.h>
int main() {
   struct utsname info;
   uname(&info);
   printf("Betriebssystem (BS) = %s\n", info.sysname);
   printf("     Release des BS = %s\n", info.release);
   printf("     Version des BS = %s\n", info.version);
   printf("Rechner-Architektur = %s\n", info.machine);
   printf("Netzwerkknoten-Name = %s\n", info.nodename);
   return 0;
}
```

In C wird jede Zahl ungleich 0 (insbesondere 1) als wahre Aussage angesehen und 0 als falsche. Diese Zuordnung ist anders als in Shellscripts (siehe unten). Neuere Versionen von C (ab C99), wie in Linux, haben einen Datentyp für Wahrheitswerte (`_Bool`). Zur Verwendung bindet man die Header-Datei `stdbool.h` ein. Damit hat man den Datentyp `bool` (ein schönerer Name für `_Bool`) mit den Werten `true`, entspricht 1,und `false`, entspricht 0. Man beachte, dass die Wahrheitswerte `true` und `false` klein geschrieben werden. Folgendes Beispiel zeigt die Verwendung von Wahrheitswerten.

```
#include <stdbool.h>
#include <stdio.h>
int main(void)
{
bool x1 = true;   bool x2 = false;
printf("x1 AND x2 = %d\n", x1 && x2); // ergibt 0
printf("x1  OR x2 = %d\n", x1 || x2); // ergibt 1
}
```

Man kann beim Aufruf eines C Programms Parameter übergeben. Sie werden vom Dateinamen und untereinander durch Leerzeichen getrennt. Zur Verarbeitung der Parameter wird die `main` Funktion mit Argumenten aufgerufen, welche traditionell `argc` und `argv` heißen. Beispiel (Programmdatei `myprog.c`, ausführbare Datei `myprog`):

```
#include <stdio.h>
int main(int argc, char *argv[]) {
   int i;
   printf("argc = %d\n",argc);
   for(i=0; i < argc; i++) {
      printf("argv[%d] = %s\n", i, argv[i]);
   }
   return 0;
}
```

Nach dem Compilieren von `myprog.c` und dem Aufruf des Programms durch

```
myprog Hallo Welt!
```

erhält man die Ausgabe

```
argc = 3
argv[0] = ./ahg01
argv[1] = Hallo
argv[2] = Welt!
```

Die ganzzahlige Variable `argc` (für argument count) enthält die Anzahl der Parameter plus 1 und `argv[0]` enthält den Pfad zur ausführbaren Datei. Die weiteren Elemente von array `argv` enthalten die übergebenen Parameter als Zeichenketten.

Die Übersetzung vom Quellcode in ein fertiges Programm (Build-Prozess) kann mit dem Programm `make` gesteuert werden. Das ist vorteilhaft, wenn der Code auf mehrere Dateien verteilt wurde. Sofern noch nicht verfügbar, wird `make` mit

```
sudo apt-get install make
```

installiert. Als einfaches Beispiel nehmen wir ein Hauptprogramm (mit der Funktion `main`) `prog.c`, welches (zusätzlich zur C Standard-Bibliothek) die Header-Dateien `foo.h` und `bar.h` braucht. Ohne `make` kann es mit dem Befehl

```
gcc -o prog prog.c
```

in eine ausführbare Datei übersetzt werden. Alternativ kann man eine besondere Datei namens `Makefile` erstellen, welche den Build-Prozess beschreibt. In unserem einfachen Beispiel könnte sie so aussehen:

```
# Build Programm für ...
prog: prog.c foo.h bar.h
        gcc -o prog prog.c
```

wobei die Einrückung der dritten Zeile durch Tabulator erfolgen muss. Zeilen, die mit dem Zeichen # beginnen, sind Kommentare. In der zweiten Zeile steht vor dem Doppelpunkt ein Ziel, das erreicht werden soll, *target* genannt. Als erstes target steht meistens der Name der ausführbaren Datei, welche erzeugt werden soll. Danach folgen, getrennt durch ein Leerzeichen, die Dateien, die dafür gebraucht werden (*dependencies* oder prerequisites genannt). Die dritte Zeile gibt den Befehl (command), mit dem das target erreicht werden soll. Bei Bedarf können weitere Befehlszeilen folgen (ebenso eingerückt mit Tabulator). Es wäre möglich, weitere targets zu definieren. Der Befehl

```
make
```

bewirkt die Ausführung des Build-Prozesses,welcher im `Makefile` beschreiben wurde. Auf der Webseite https://www.gnu.org/software/make/manual/ findet man die Dokumentation zu `make` und auf https://makefiletutorial.com/ ein Tutorial.

Kontrollstrukturen

Ein einfacher if-Befehl enthält eine auswertbare Bedingung, wahr oder falsch sein kann. Genau dann, wenn sie wahr ist, wir der unmittelbar folgende Befehl oder ein (in geschweiften Klammern gesetzter) Block (aus einem oder mehreren Befehlen) ausgeführt.

```
#include <stdio.h>
int main() {
int n = 13;
if (n == 13) printf("n ist 13\n");
if (n != 13)
{ printf("n ist nicht 13\n"); }
return 0; }
```

Ist die Bedingung falsch, kann mit `else` ein alternativer Befehl ausgeführt werden.

```
#include <stdio.h>
int main() {   int n = 13;
if (n <= 10) printf("n ist hoechstens 10\n");
else  printf("n ist groesser als 10\n");   return 0; }
```

Mehrere Möglichkeiten gibt es mit einem oder mehreren `else if`.

```
#include <stdio.h>
int main() {   char  bs[3] = {'a', 'b', 'c'};
if      (bs[1] == 'a') printf("der Buchstabe ist a\n");
else if (bs[1] == 'b') printf("der Buchstabe ist b\n");
else if (bs[1] == 'c') printf("der Buchstabe ist c\n");
else printf("der Buchstabe ist weder a noch b noch c\n");
return 0; }    // der else-Zweig ist optional
```

Mehrere Möglichkeiten für eine ganzzahlige Variable oder einen ganzzahligen Ausdruck können auch mit dem `switch` Befehl ausgewertet werden, der auf Gleichheit prüft. Außer im Ersatzfall `default` wird jeder Fall mit einem `break` Befehl abgeschlossen.

```
#include <stdio.h>
int main() {   int n = 8;
switch(n) {   // geprüft wird, ob n gleich 1, 2 oder 3 ist
    case 1: printf(" erster Fall\n"); break;
    case 2: printf("zweiter Fall\n"); break;
    case 3: printf("dritter Fall\n"); break;
   default: printf("anderer Fall\n"); }
return 0;   }
```

Eine `for`-Schleife beginnt mit `for` und einem Klammerausdruck, dessen Teile durch Semikola getrennt werden. Der erste Teil ist ein einmalig zu Beginn ausgeführter Befehl,

welcher in der Regel eine Laufvariable initialisiert. Der zweite Teil ist ein Ausdruck, der zu wahr oder falsch ausgewertet wird. Wahr kann auch durch eine Zahl $\neq 0$ dargestellt werden, meistens 1, und falsch durch 0. Der dritte Teil ist ein Befehl, der am Ende jedes Schleifendurchlaufs ausgeführt wird. Genau dann, wenn der zweite Teil wahr ist, wird der auf den Klammerausdruck folgende Befehl oder Befehlsblock ausgeführt.

```
#include <stdio.h>
int main() {   int i;
for( i = 1; i <= 10; i++)
   printf("i: %2d\n", i);
return 0;   }
```

Von der `for`-Bedingung muss mindestens ein Befehl abhängen, notfalls ein leerer (;).

Eine `while`-Schleife wird so oft durchlaufen, wie die Bedingung erfüllt ist.

```
#include <stdio.h>
int main() {   int i = 1;
while ( i <= 10 )
   { printf("i: %2d\n", i); i++; }
return 0;   }
```

Wenn die Bedingung zu Beginn nicht erfüllt wird, dann wird die Schleife nie durchlaufen.

Der Befehl `break` im Schleifenblock beendet den Durchlauf. Im folgenden Beispiel zählt die Schleife nur bis 2, obwohl die Bedingung wahr ist (Abbruch einer Endlosschleife).

```
#include <stdio.h>
int main() {   int i = 1;
while ( 1 )
   { printf("i: %2d\n", i); i++;
     if ( i==3 ) break; }
return 0;   }
```

Eine Schleife mit `do` und `while` wird mindestens einmal durchlaufen. Im folgenden Beispiel werden ganze Zahlen von der Tastatur eingelesen, bis eine 0 eingegeben wird. Ist `inp` = 0, ist die Bedingung nach `while` falsch. Die Summe wird ausgegeben.

```
#include <stdio.h>
int main() {   int inp = 0, sum ;
do        // läuft mindestens einmal
{ scanf("%d", &inp) ; sum += inp ; }
while ( inp ) ;        // Ende mit 0
printf("Summe: %d\n", sum) ; return 0 ;   }
```

Man beachte, dass nach der `while`-Bedingung ein Semikolon den Befehl abschließt.

3.4.2 Beispiele mit zusätzlichen Header-Dateien

Will man Dateien einbinden, die nicht in der C Standard-Bibliothek enthalten sind, zum Beispiel `json-c/json.h` und `curl/curl.h`, muss man sie erst installieren.

```
sudo apt-get install libcurl-dev
sudo apt-get install libjson-c-dev
```

Folgendes Programm liest die öffentliche IP-Adresse unseres Rechners (siehe Seite 153) im Datenformat JSON (siehe Seite 303) von einem frei zugänglichen Server, speichert sie in der Datei `pip.json` und gibt sie auf dem Bildschirm aus.

```
// --- Programm zur Augabe der oeffentlichen IP-Adresse ---
#include <json-c/json.h>
#include <curl/curl.h>
int main () {
// --- Download der JSON-Datei ---
   char *url_json = "https://api.ipify.org/?format=json";
   FILE *pip_file = fopen("pip.json", "w");
   CURL *curl_handle = curl_easy_init();
   curl_easy_setopt(curl_handle, CURLOPT_URL, url_json);
   curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION, fwrite);
   curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, pip_file);
   curl_easy_perform(curl_handle);
   curl_easy_cleanup(curl_handle);
   fclose(pip_file);
// --- Einlesen der JSON-Datei ---
   char buffer[1024];
   pip_file = fopen("pip.json", "r");
   fread(buffer, 1024, 1, pip_file);
   fclose(pip_file);
// --- Parsing der JSON-Datei ---
   struct json_object *parsed_json;
   struct json_object *pip;
   parsed_json = json_tokener_parse(buffer);
   json_object_object_get_ex(parsed_json, "ip", &pip);
   printf("Public IP address: %s\n", json_object_get_string(pip));
   return 0;
}
```

Compilierung des in der Datei `pip.c` gespeicherten Programms erfolgt durch den Befehl

```
gcc -o pip pip.c -lcurl -ljson-c
```

und zur Ausführung genügt dann

```
./pip
```

Folgendes Programm namens `euro.c` gibt den Wechselkurs des Euro zum chinesischen Yuan und zum US-Dollar auf dem Bildschirm aus. Die aktuellen Wechselkurse werden im JSON Datenformat von einem Server bereitgestellt, ohne dass es einer Anmeldung bedarf. Die Daten verschachtelt: Die Währungsschlüssel "CNY" und "USD" befinden sich in einem Teilobjekt, welches der Wert des Schlüssels "rates" ist.

```
// --- Programm zur Augabe aktueller Euro-Wechselkurse ---
#include <curl/curl.h>
#include <json-c/json.h>
int main () {
    // --- Download der JSON-Datei ---
   char *url_json = "https://open.er-api.com/v6/latest/EUR";
   FILE *fil = fopen("pip.json", "w");
   CURL *curl_handle = curl_easy_init();
   curl_easy_setopt(curl_handle, CURLOPT_URL, url_json);
   curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION, fwrite);
   curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, fil);
   curl_easy_perform(curl_handle);
   curl_easy_cleanup(curl_handle);
   fclose(fil);
   // --- Einlesen der JSON-Datei ---
   char buffer[10240];
   fil = fopen("pip.json", "r");
   fread(buffer, 10240, 1, fil);
   fclose(fil);
   // --- Parsing der JSON-Datei ---
   struct json_object *parsed_json;
   struct json_object *rat;
   struct json_object *cny;
   struct json_object *usd;
   size_t n_rates;
   parsed_json = json_tokener_parse(buffer);
   json_object_object_get_ex (parsed_json, "rates", &rat);
   json_object_object_get_ex(rat, "CNY", &cny);
   printf("1 Euro = %s chines. Yuan\n", json_object_get_string(cny));
   json_object_object_get_ex(rat, "USD", &usd);
   printf("1 Euro = %s $ (US-Dollar)\n", json_object_get_string(usd));
   return 0;
}
```

Folgendes Programm namens `bild.c` lädt ein Bild aus dem Internet und speichert es in einer Bilddatei (Download). Der Name der Bilddatei enthält einen Zeitstempel, welcher aus Datum und Zeit des Downloads besteht. Damit können verschiedene Bilder gespeichert werden. Außerdem bewirkt das Format des Zeitstempels, dass bei den Dateinamen verschiedener Downloads die alphabetische Reihenfolge gleich der chronologischen ist.

```
#include <locale.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <curl/curl.h>
int main() {

char *locale;
locale = setlocale(LC_ALL, "");
char now[256]; time_t t = time(NULL);
char out[80];           // image file, name
strcpy(out, "cartoon"); // image file, base name
char dtt[15];
struct tm * tim = localtime(&t);
strftime(dtt, sizeof(dtt), "_%y%m%d_%H%M%S", tim);
strcat(out,dtt);        // image file, time
strcat(out,".png");     // image file, extension

char url[] = "https://imgs.xkcd.com/comics/presents_for_biologists.png";
CURL *hdl;
CURLcode imgresult;
FILE *fil = fopen(out, "wb");
hdl = curl_easy_init();
curl_easy_setopt(hdl, CURLOPT_URL, url);
curl_easy_setopt(hdl, CURLOPT_WRITEFUNCTION, NULL);
curl_easy_setopt(hdl, CURLOPT_WRITEDATA, fil);
imgresult = curl_easy_perform(hdl);
if(imgresult) printf("Error in download\n");
curl_easy_cleanup(hdl);
fclose(fil);

return 0;
}
```

Auf Fehlerbehandlung wird weitgehend verzichtet. Das Programm wird mit

```
gcc -o bild bild.c -lcurl
```

compiliert. Zur Ausführung gibt man den Befehl

```
./bild
```

ein. Die Datei `cartoon.png` speichert das Bild, welches heruntergeladen wurde. Es kann mit einem Bildbetrachtungsprogramm wie `fbi` (siehe Seite 173) angesehen werden.

Die Grafikbibliothek DISLIN von Helmut Michels ermöglicht die Darstellung wissenschaftlicher Daten. Sie ist kann mit dem gcc Compiler des Raspberry Pi oder mit Python

genutzt werden und ist frei verfügbar für nicht-kommerzielle Anwendungen. Es gibt eine Anleitung für C und C++ und einedislin für Python, sowie zahlreiche Beispiele für verschiedene Programmiersprachen.

Man kann Quellcode in eine Header-Datei (die üblicherweise die Endung `.h` erhält) auslagern, damit der Code übersichtlicher wird. Im folgenden Beispiel wird die Definition einer Funktion `wf` in eine Header-Datei `f_welt.h` geschrieben,

```
const char* wf() {
   return "Welt!";
}
```

während das Hauptprogramm mit der `main`-Funktion in der Datei `hw.c` steht.

```
#include <stdio.h>
#include "f_welt.h" // definiert wf
int main () {
    const char* name = wf();
    printf("Hallo %s\n", name);
   return 0;
}
```

Der `include`-Befehl kopiert den Inhalt der Header-Datei an seiner Stelle in das Hauptprogramm. Der Name der Header-Datei kann in spitzen Klammern oder in doppelten Anführungszeichen eingefasst sein. Je nach Schreibweise sucht der Compiler die Header-Datei in verschiedenen Verzeichnissen und tut dies in bestimmter Reihenfolge.

Header-Dateien zur Auslagerung von Quellcode stehen in der Regel in doppelten Anführungszeichen und werden daher zunächst im selben Verzeichnis gesucht wie die Datei, welche das Hauptprogramm (mit dem `include`-Befehl) enthält.

4 Die Shell

Die Kommunikation eines Nutzers mit dem Betriebssystem des Rechners erfolgt über eine Schnittstelle (interface). Es gibt zwei Arten dieser Schnittstellen:

- CLI = command-line interface (Kommandozeile oder Befehlszeile)
- GUI = graphical user interface (graphische Benutzeroberfläche)

In Linux, als Abkömmling des Betriebssystems UNIX, besteht das CLI aus dem Terminal und einem besonderen Computerprogramm, der UNIX-Shell.

Die UNIX-Shell ist ein Programm, das Befehle des Nutzers (user) in Textform über die Tastatur annimmt, aufbereitet und an den Kern des Betriebssystems (Kernel) weitergibt. Es gibt für Linux verschiedene UNIX-Shells. Wir verwenden die `bash` (Bourne-again-shell, benannt nach Erfinder der Vorversion `bsh`, Stephen Bourne).

In Linux-Systemen mit grafischer Benutzeroberfläche wird für die UNIX-Shell ein „Terminalfenster“ auf dem Bildschirm geöffnet. Im Betriebssystem Windows gibt es eine Entsprechung, die „Eingabeaufforderung“.

Wir nennen eine UNIX-Shell meistens kurz Shell, obwohl man den Begriff Shell allgemeiner definieren kann.[1] Eine UNIX-Shell vermittelt aber nicht nur zwischen Nutzer und Rechner. Sie führt außerdem Befehle einer eigenen Programmiersprache aus.

Um zu sehen, welche UNIX-Shells verfügbar sind, gibt man den Befehl

```
cat /etc/shells
```

und erhält zum Beispiel im Betriebssystem `Raspberry OS lite`, Version vom 4. April 2022, die Antwort, dass in den Verzeichnissen `/bin` und `/usr/bin` die Shells `bash`, `dash`, `mksh` und `mksh-static` (eine Variante von `mksh`) verfügbar sind, siehe Tabelle 4.1.

Shell	Langform ds Namens	Kommentar
`bash`	Bourne-again shell	häufig verwendete Shell des GNU Projekts
`dash`	Debian Almquist shell	kleine, schnelle Shell in Debian und Ubuntu
`mksh`	MirBSD Korn Shell	Nachfolger der Korn Shell (Standard in Android)

Tabelle 4.1: Bereits installierte Shell-Programme in `Raspberry OS lite` (4. April 2022), ohne die für einen schnellen Systemstart optimierte Variante `mksh-static`

Außerdem gibt es Verweise von `rbash` auf `bash` und von `sh` auf `dash`. Das Verzeichnis `/bin` gibt es aber eigentlich nicht. Es ist nur ein Verweis (symlink) auf das Verzeichnis

[1] Im Englischen bedeutet `shell` Schale, Muschelschale oder Hülle. Bildlich gesprochen, umhüllt die Shell den Kern des Betriebssystems (Kernel). Der Begriff *graphical shell* beschreibt die Desktop-Umgebung des Betriebssystems `Raspberry Pi OS with desktop`.

`/usr/bin`. In gleicher Weise sind `/sbin` ein Verweis auf `/usr/sbin` und `/lib` ein Verweis auf `/usr/lib`. Der Grund für diese umständliche Konstruktion liegt in der geschichtlichen Entwicklung von Linux.

Der direkte Pfad zu einer Datei oder zu einem Verzeichnis, ohne Verweise und ohne andere Namen (wie `.` für das aktuelle Arbeitsverzeichnis oder `..` für das übergeordnete Verzeichnis) und ohne mehrfache (also überflüssige) Schrägstriche `/`, heißt *kanonischer Pfad* (canonical path). Den kanonischen Pfad zu einem Pfad `X` erhält man mit

```
realpath X
```

wobei die Antwort aber nicht nur für existierende Pfade `X`, sondern auch für nicht existierende Pfade gegeben wird. Letztere könnte man ja erzeugen. In obigem Beispiel gibt es also nur zwei verfügbare Shell-Programme mit den kanonischen Pfaden `/usr/bin/bash` und `/usr/bin/dash`.

Ein laufendes Shell-Programm ist ein Shell-Prozess (oft auch nur Shell genannt). Es können mehrere Shell-Prozesse gleichzeitig laufen. Ein Shell-Prozess kann interaktiv arbeiten oder nicht und er kann eine Login-Shell sein oder nicht. Daraus ergeben sich vier Betriebsarten (Modi):

- interaktiver Login-Shell-Prozess
- nicht-interaktiver Login-Shell-Prozess
- nicht-interaktiver Non-Login-Shell-Prozess
- interaktiver Non-Login-Shell-Prozess

Ein interaktiver Shell-Prozess nimmt Befehle des Nutzers entgegen, die in Textform über die Tastatur eingegeben werden. In der Regel erfolgt eine Ausgabe von Text auf dem Bildschirm. Die Shell, mit der wir nach dem Systemstart in der Konsole arbeiten, ist interaktiv. Eine Shell kann aber auch Befehle abarbeiten, die als sogenanntes *Shellscript* in einer Datei gespeichert sind. Das macht eine nicht-interaktive Shell.

Beim Hochfahren des Betriebssystems wird nach der Anmeldung (login) eines Nutzers (user) in der Konsole (ohne graphische Benutzeroberfläche) ein Login-Shell-Prozess gestartet. Danach können (durch den Nutzer oder durch andere Prozesse) weitere Shell-Prozesse gestartet werden. Diese sind in der Regel Non-Login-Shells, außer wenn die Login-Eigenschaft ausdrücklich gewünscht wird. Letzteres kann mit der Option `- -login` geschehen. Übrigens sind die Shell-Prozesse, die in einem Betriebssystem mit graphischer Benutzeroberfläche beim Öffnen eines Terminalfensters gestartet werden, in der Regel interaktive Non-Login-Shells. Jeder Shell-Prozess führt nach seinem Start einige Einrichtungen durch, die in Konfigurationsdateien beschrieben werden. Eine Login-Shell macht aber mehr als eine Non-Login-Shell, weil einige Einrichtungen nur nach dem Anmelden (login) des Nutzers einmalig erforderlich sind.

Welche Konfigurationsdateien beim Start einer Shell gelesen und ausgeführt werden, hängt von vielen verschiedenen Dingen ab. Wir betrachten die beiden wichtigsten Fälle beim Start einer `bash` in unserem Betriebssystem (ohne graphische Benutzeroberfläche). Außerdem schauen wir, was in einem Betriebssystem mit graphischer Benutzeroberfläche geschieht, wenn ein Terminalfenster geöffnet wird und damit eine bash gestartet wird. Wie üblich ist $\sim$ ein Kürzel für das Heimatverzeichnis des aktuellen Nutzers.

1) Eine interaktive Login-bash (bei der wir normalerweise ohne graphische Benutzeroberfläche im Terminal Befehle eingeben) führt (unabhängig davon, welcher Nutzer sich angemeldet hat) das Script `/etc/profile` aus. Dieses ruft wiederum einige Scripte auf, die im Verzeichnis `/etc/profile.d` stehen, und auf die wir hier nicht eingehen. Danach werden, der Reihe nach, die Dateien `~/.bash_profile`, `~/.bash_login` und `~/.profile` gesucht und nur die erste gefundene Datei ausgeführt. Wir verzichten auf die beiden ersten dieser drei Dateien und benutzen nur `~/.profile`. Das darin enthaltene Script führt zunächst die Datei `~/.bashrc` aus. Danach werden weitere Einstellungen vorgenommen. Wichtig: Am Ende des Scripts können wir eigene Befehle anfügen, die nach dem Anmelden (login) automatisch einmalig für den angemeldeten Nutzer ausgeführt werden sollen. Da auch `~/.bashrc` ausgeführt wird, brauchen wir Befehle, die in der Datei `~/.bashrc` stehen, nicht nochmal in `~/.profile` schreiben.

2) Eine nicht-interaktive Non-Login-bash wird benutzt, wenn ein Shellscript ausgeführt wird, das in einer Datei gespeichert ist. Nach dem Start liest die bash den Wert der Umgebungsvariablen `BASH_ENV`. Wenn der Wert den Namen einer ausführbaren Datei ergibt, wird diese ausgeführt. Andere Konfigurationsdateien werden nicht aufgerufen. Solange man `BASH_ENV` keinen Wert zuweist, wird also beim Start der nicht-interaktiven Non-Login-bash keine Konfigurationsdatei ausgeführt (auch nicht `~/.bashrc`).

3) Eine interaktive Non-Login-bash kann in unserem Betriebssystem (ohne graphische Benutzeroberfläche) mit dem Befehl `bash` gestartet werden und mit `exit` verlassen werden (mit Rückkehr zur interaktiven Login-bash). Das ist aber kaum nützlich. In einem Betriebssystem mit graphischer Benutzeroberfläche wird eine interaktive Non-Login-bash gestartet, wenn ein Terminalfenster geöffnet wird. Eine interaktive Non-Login-bash führt das Script `~/.bashrc` aus, nicht jedoch `~/.profile`.

In einem Betriebssystem ohne graphische Benutzeroberfläche kann man eigene Konfigurationsbefehle in die Datei `~/.profile` oder in die `~/.bashrc` schreiben. Beide Dateien werden normalerweise bei der Anmeldung (login) ausgeführt.

Mit einer graphischen Benutzeroberfläche wird dagegen beim Start des Betriebssystems durch den DisplayManager einmalig die Datei `~/.profile` ausgeführt und bei jedem Öffnen eines Terminalfensters und Start der bash die Datei `~/.bashrc` ausgeführt. Grundsätzlich sollte man eigene Konfigurationsbefehle so auf die Dateien `~/.profile` und `~/.bashrc` verteilen, dass sich ohne und mit graphischer Benutzeroberfläche das gewünschte Verhalten ergibt.

Es ist wichtig zu wissen, mit welcher UNIX-Shell im Terminal interaktiv gearbeitet wird und welche UNIX-Shell bei der Abarbeitung von Shellscripts verwendet wird. Ohne besondere Festlegung seitens des Nutzers arbeiten die Betriebssysteme `Debian` und `Raspberry OS` mit der `bash` als interaktiver Shell. Für Shellscripts, die vom Betriebssystem gestartet werden, wird jedoch (seit der Debian Version `Squeeze`) die `dash` eingesetzt. Ubuntu verhält sich ähnlich wie Debian.

Leider unterscheiden sich verschiedene Linux-Betriebssysteme in den Standards für die Verwendung einer Shell. Es ist daher sinnvoll, eigene Shellscripts so zu schreiben, dass sie mit vielen Linux-Betriebssystemen und mit verschiedenen Schnittstellen (CLI / GUI) funktionieren. Dafür sollte man dem einfachen POSIX Standard folgen, aber andererseits

will man oft auf die erweiterten Möglichkeiten der `bash` nicht verzichten.

Aus den oben genannten Gründen ist es bei Verwendung der `bash` (wie in diesem Manuskript) sinnvoll, in jedem Shellscript eine besondere erste Zeile einzufügen, die dem Betriebssystem mitteilt, dass als Shell die `bash` verwendet werden soll. Diese besondere erste Zeile beginnt mit dem *Shebang* `#!`. Sie kann so aussehen:

```
#!/usr/bin/env bash
```

Erläuterungen zur Syntax der Shebang-Zeile findet man im Internet.

Shellscripte kann man auch in Linux-Betriebssystemen mit GUI (graphical user interface) einsetzen, zum Beispiel im `Raspberry Pi OS with desktop`. Wenn ein Shellscript keine Shebang-Zeile hat und in einem Terminal aufgerufen wird, wird das Shellscript im `Raspberry Pi OS with desktop` standardmäßig mit der `bash` ausgeführt. Wird das gleiche Shellscript jedoch durch einen `desktop shortcut` gestartet, wird es mit der `dash` ausgeführt und ergibt vielleicht eine Fehlermeldung. Wenn dagegen das Shellscript die Shebang-Zeile `#!/usr/bin/env bash` enthält, wird es mit der `bash` ausgeführt.

4.1 Befehle für System und Shell

Texteingaben im Terminal werden von der UNIX-Shell, kurz Shell genannt, ausgewertet. Sie arbeitet dann als command-line interpreter (Kommandozeile). Der Nutzer gibt eine Zeichenkette ein und schließt sie mit der Enter-Taste ab. Die Zeichenkette wird von der Shell interpretiert. Wird die Eingabe nicht als Befehl der Shell erkannt, wird geprüft, ob es einen Systembefehl (ein zum Betriebssystem gehörendes Programm) oder ein Anwendungsprogramm mit dem eingegebenen Namen gibt. Wir fassen hier die Befehle der Shell und Systembefehle zusammen.

4.1.1 Arbeit mit der Bash

Wer gerne aufräumt, erhält mit der Tastenkombination `Ctrl-l` (mit l wie leer), oder mit dem Befehl `clear`, eine Kommandozeile auf einem ansonsten leeren Bildschirm.

history

Die Bash merkt sich Befehlszeilen, die in Bash-Sitzungen eingegeben (und mit der ENTER-Taste abgeschlossen) wurden, in einer Liste, der *history*. Mit dem Befehl

```
history | less
```

wird die history als nummerierte Liste auf dem Bildschirm ausgegeben. Die Verwendung von `less` (ein terminal pager) ist sinnvoll, weil die Liste viele Einträge enthält (zum Beispiel 1000, abhängig von den Einstellungen, siehe unten). `history` ist ein Befehl der Shell `bash`, den es auch bei vielen anderen, aber nicht bei allen Shells gibt.

Erneutes Eintippen alter Befehle ist unbequem. Die Liste kann daher nützlich sein, wenn man einen früher verwendeten, umfangreichen Befehl erneut braucht. Durch

```
!1997
```

kann zum Beispiel der Befehl Nummer 1997 erneut aufgerufen werden. Das geht jedoch nur direkt nach dem Anschauen der Liste, da die history ja fortlaufend aktualisiert wird.

Mit den Pfeiltasten ↑ und ↓ kann man die *history* durchsuchen und einen Befehl dann mit der ENTER-Taste nochmals geben.

In der Datei ∼/.bash_history wurden maximal n_{alt} Befehlszeilen aus früheren Bash-Sitzungen gespeichert. Hinzu kommen die Befehlszeilen der aktuellen Bash-Sitzung. Die Anzahl der Befehlszeilen in der *history* wird jedoch auf maximal n beschränkt.

Mit den Befehlen

```
history -c
history -w
```

werden der Inhalt der *history* und der Inhalt der Datei ∼/.bash_history gelöscht.

Das Verhalten der *history* wird in der Datei ∼/.bashrc gesteuert. In der Zeile

```
HISTSIZE=1000
```

ist der Wert von n bestimmt. Entsprechend ist in

```
HISTFILESIZE=2000
```

der Wert von n_{alt} bestimmt. Beide Werte können geändert werden.

Meistens gibt es eine Zeile HISTCONTROL=ignoreboth, die dafür sorgt, dass a) Befehlszeilen, die mit Leerzeichen beginnen, nicht in die *history* aufgenommen werden, und b) aufeinander folgende gleiche Befehlszeilen nur einmal in die *history* aufgenommen werden. Setzt man stattdessen

```
HISTCONTROL=ignoreboth:erasedups
```

werden außerdem alle früheren Doppel (nicht nur die unmittelbaren Vorgänger) gelöscht. Will man verhindern, dass eine Befehlszeile in die *history* aufgenommen wird, beginnt man sie mit einem Leerzeichen.

Namensergänzung mit Tab

Die Namen von Befehlen oder Dateien werden automatisch vervollständigt, wenn nach den ersten Zeichen die Tabulatortaste Tab gedrückt wird und die Ergänzung eindeutig ist. Wenn die Ergänzung nicht eindeutig ist, wird soweit ergänzt, wie alle Möglichkeiten übereinstimmen. Man kann dann entweder selbst den Namen vervollständigen oder nochmal Tab drücken um die Möglichkeiten anzeigen zu lassen.

Ergänzung von Dateinamen durch globbing

In Befehlszeilen werden oft Dateien angegeben. Mitunter braucht man nicht einen Dateinamen, sondern eine Menge ähnlicher Dateinamen, die durch ein Muster beschrieben werden können. Dies geschieht durch *globbing*, benannt nach dem früher dafür benutzen Programm namens glob. Wenn das Muster durch ein einzelnes Zeichen dargestellt wird, nennt man es auch *Wildcard*. Für Dateinamen gibt es in der Bash folgende Muster:

? genau ein Zeichen

***** beliebig viele Zeichen

{*zk1*,*zk2*} Zeichenkette *zk1* oder Zeichenkette *zk2*

[*abc*] genau eines der einzelnen Zeichen *a*,*b*,*c*

Mit dem Befehl

```
ls [hH]*.{jpg,png} 2> /dev/null
```

werden alle Dateinamen ausgegeben, die mit `h` oder `H` beginnen und mit `.jpg` oder `.png` enden. Durch `2> /dev/null` werden Fehlermeldungen unterdrückt, wenn zum Beispiel keine entsprechende Datei mit der Endung `png` gefunden wurde.

Man beachte, dass die Wildcard $*$ bei der Dateinamen-Expansion keine Zeichenketten mit einem Punkt am Anfang repräsentiert. Zum Beispiel werden mit dem Befehl

```
ls *.[pP][nN][gG]
```

alle Dateien mit der Endung `.png` ausgegeben, unabhängig von Groß- oder Kleinschreibung der Buchstaben in der Endung des Dateinamens, aber ohne Dateien, deren Name mit einem Punkt beginnt (wie etwa `.hallo.png`).

copy & paste in der bash

Innerhalb der bash gibt es copy & paste. Durch die Tastenkombination `Ctrl-w` (gleichzeitiges Drücken der Tasten `Ctrl` und `w`) wird das Wort links vom Cursor aus der Kommandozeile ausgeschnitten und in einen Zwischenspeicher der bash übertragen. Entsprechend wird der Inhalt des Zwischenspeichers mit `Ctrl-w` an der Stelle des Cursors eingefügt. Zum Kopieren führt man `Ctrl-w` und `Ctrl-w` hintereinander aus. Der Zwischenspeicher der bash wird leider nicht von anderen Programmen, wie Texteditoren, erkannt.

4.1.2 Dateien, Netzwerk, Prozesse, Rechnen

Über die Wahl geeigneter Dateinamen wurde bereits gesprochen (siehe Seite 92).

Information zum Dateisystem

`ls`	Ausgabe des Inhalts vom aktuellen Verzeichnis	`ls`
	einschließlich versteckter Dateien, in Langform	`ls -al`
	sortiert nach Änderung (mtime), in Langform	`ls -tl`
	einzeilige Ausgabe eines Verzeichnis-Inhalts	`ls -m` *verz*
`file`	Ausgabe: Dateityp (anhand Inhalt), Kurzform	`file -b` *dat*
	für spezielle Datei (Dateisystem eines device)	`file -bs` *dat*
`find`	Dateisystemsuche nach Dateiname (groß/klein)	`find / -iname` *name*
`realpath`	Pfad zur Datei (`-e` : Pfad muss existieren)	`realpath -e` *dat*
`basename`	Ausgabe des Dateinamens eines Dateipfads	`basename` *dat*
`dirname`	und des Verzeichnisnamens eines Dateipfads	`dirname` *dat*

Arbeit mit dem Dateisystem

`cd`	Wechsel in ein Unterverzeichnis des aktuellen V.	`cd` *uverz*
	Wechsel in das übergeordnete Verzeichnis	`cd ..`
	Verzeichniswechsel (Leerzeichen sind möglich)	`cd /`*verz*`/"`*uverz*`"`

Das Einfassen in Anführungsstrichen ermöglicht Leerzeichen im Namen *uverz* .

`cp`	Datei kopieren (überschreibt vorhandenes Ziel)	`cp` *dat1 dat2*
	Datei in das aktuelle Verzeichnis (.) kopieren	`cp` *dat1* .
	Verzeichnis rekursiv kopieren (mit Inhalt)	`cp -r` *verz1 verz2*
	mehrere Dateien in ein Verzeichnis kopieren	`cp` *dat1 dat2 verz*
	kopieren, wenn Quelle neuer als Zieldatei ist	`cp -u` *dat1 verz*
	ebenso mit Attributen (Zugriffsrechte usw.)	`cp -pu` *dat1 verz*
	ebenso, kopiert aber bei Softlink nicht die Datei	`cp -dpu` *dat1 verz*
`dd`	Datei kopieren, Formatänderungen sind möglich	`dd if=`*dat1* `of=`*dat2*
`mv`	Datei umbenennen (alt: *dat1*, neu: *dat2*)	`mv` *dat1 dat2*
	Datei verschieben (entsprechend: Verzeichnis)	`mv` *dat verz*
	ebenso, wenn Quelle neuer als Ziel (u = update)	`mv u` *dat verz*
	ebenso, und immer ohne Nachfrage (f = force)	`mv fu` *dat verz*
`rm`	Datei löschen ohne Nachfrage, auch read-only	`rm -f` *dat*
	Verzeichnis mit Inhalt ohne Nachfrage löschen	`rm -fr` *verz*
`touch`	Erzeugung einer leeren Datei	`touch` *dat*
`ln`	Softlink (symbolische Verknüpfung) auf Datei	`ln -s` *dat slk*

	Hardlink (Verweis auf physikalische Adresse)	`ln` *dat hlk*
`mkfifo`	Erzeugung einer FIFO-Datei (named pipe)	`mkfifo` *dat*
`chown`	Besitzwechsel des Verzeichnisses (R = recursive)	`chown -R` *name verz*
`chgrp`	Gruppenwechsel der Datei (nur für Mitglieder)	`chgrp` *gruppe dat*
`chmod`	Änderung der Zugriffsrechte auf eine Datei	`chmod` *rechte dat*
	Datei für alle (a = all) ausführbar (x = execute)	`chmod a+x` *dat*

Archivierung von Dateien

`cksum`	Prüfsumme, Anzahl Bytes und der Dateiname	`cksum` *dat*
`md5sum`	Ausgabe der MD5-Prüfsummen-Zeichenkette	`md5sum` *dat*
`sha256sum`	entsprechend für die SHA256-Prüfsumme	`sha256sum` *dat*
`tar`	packe Verzeichnis unkomprimiert in Archivdatei	`tar -cf` *dat.tar* `-C` *verz* `.`
	(`.` am Ende beachten); entpacke die Archivdatei	`tar -xf` *dat.tar*
	Verzeichnis packen und mit gzip komprimieren	`tar -czf` *dat.tar.gz* `-C` *verz* `.`
	im aktuellen V. dekomprimieren und extrahieren	`tar -xzf` *dat.tar.gz*
	Vorhandene Dateien werden beim Extrahieren (Entpacken) überschrieben!	
`xz`	komprimiere eine Datei und lösche das Original	`xz` *dat*
	(kleines Ziel, langsam); entpacke eine xz-Datei	`unxz` *dat.xz*
`zip`	packe ein Verzeichnis komprimiert (r = recursive)	`zip -r` *dat.zip verz*
	(Windows und Linux); entpacke eine zip-Datei	`unzip` *dat.zip*

Arbeit mit Datei-Inhalten

`cat`	Bildschirm-Ausgabe des Inhalts der Datei *dat*	`cat` *dat*
	Ausgabe der Datei mit Zeilen-Nummern	`cat -n` *dat*
	Konkatenation (Zusammenfügen) von Dateien	`cat` *dat1 dat2* > *dat3*
`cut`	Entnahme von Feld *n* aus allen Dateispalten	`cut -f` *n dat*
	ebenso, aber mit Feldtrennzeichen `:` statt Tab	`cut -f` *n* `-d ':'` *dat*
`diff`	Unterschiede in Dateien (`-q` ohne Einzelheiten)	`diff -q` *dat1 dat2*
`expand`	ersetze jeden Tab durch *n* Leerzeichen	`expand -t` *n dat*
`fmt`	Formatierung einer Textdatei, 75/Zeile (default)	`fmt -w 75` *dat*
	Kurzzeilen bleiben (`s`), Wort-/Satzabstand (`u`)	`fmt -su` *dat*
`head`	Ausgabe der ersten *z* Zeilen einer Datei	`head -n` *z dat*

`paste` Konkatenation entsprechender Zeilen, mit Tab `paste` *dat1 dat2*

`pdftotext` speichert Text einer PDF-Datei in einer Textdatei, siehe Seite 180

`pdfseparate` teilt eine mehrseitige PDF-Datei in Einzelseiten, siehe Seite 180

`shuf` Bildschirm-Ausgabe einer zufälligen Datei-Zeile `shuf -n 1` *dat*
Zufallszahl zwischen *a* und *b* (einschließlich) `shuf -i` *a b*
zufällige Permutation der Zeilen in einer Datei `shuf -o` *dat* `<` *dat*

`sort` Ausgabe der alphabetisch sortierten Dateizeilen `sort` *dat*
nach Sortierung: Speicherung in gleicher Datei `sort -o` *dat dat*
Sortierung ohne gleiche Zeilen speichern in *dat2* `sort +u` *dat1* > *dat2*
Sortierung der Dateizeilen ab dem n+1. Feld `sort +`n *dat*
zufällige Reihenfolge verschiedener Dateizeilen `sort -R` *dat*

`tail` Ausgabe der letzten z Zeilen einer Datei `tail -n` z *dat*

`tee` Duplikation der Ausgabe (ersetzt Datei-Inhalt) *befehl* `tee |` *dat*
Duplikation ergänzt Datei-Inhalt (a = append) *befehl* `tee -a |` *dat*

`tr` Ersetzung aller Zeichen $a \to b$, $c \to d$ in *dat1* `tr [ac] [bd]` <*dat1* >*dat2*
Ausgabe von *dat* ohne nicht druckbare Zeichen `tr -cd [:print:]` < *dat*

`wc` Anzahl Zeilen von *dat* mit \n am Zeilenende `wc -l` *dat*
Anzahl der Zeichen (`-m`) oder Bytes (`-c`) `wc -m` *dat*

Netzwerk

`curl` speichert Internet-resource in gleichnamiger Datei `curl -O` *dat*
`curl` nutzt die Bibliothek `libcurl`; `HTTP`, `HTTPS`, `FTP`, `RTMP` und mehr

`ping` prüft Internetverbindung; Option `-4` für IPv4 `ping -4 google.com`
Option `-6` für IPv6; Option `-c1` nur einmal `ping -6 -c1 google.com`

`traceroute` Weg durch das Internet zu einer URL `traceroute google.com`

`wget` lädt Internet-resource einfach als Datei herunter `wget` *url*
`HTTP`, `HTTPS`, `FTP`; Option `-i` für Datei mit URLs `wget -i` *dat*

Arbeiten mit Prozessen

`at` Ausführung von *dat* mit `dash` zu bestimmter Zeit `at 23:55 -f` *dat*
Ausgabe in `/var/mail/pi`, falls nicht mit `> /dev/tty` umgeleitet
Installation mit `sudo apt install at`, siehe auch `atq`, `atrm`, `crontab`

`atq`	Anzeige der mittels `at` auszuführenden Jobs	`atq`
`atrm`	Löschen des mittels `at` auszuführenden Jobs *n*	`atrm` *n*
`flock`	*prg* nur einmal ausführen, *dat* ist beliebige Datei	`flock -n` *dat prg*
`free`	RAM (Arbeitsspeicher) der CPU (ohne GPU) in Kibibyte = 1024 Byte "gesamt" = "benutzt" + "frei" + "Puffer/Cache", "P./C." ist teilverfügbar	
`kill`	Signal 15 (bitte beenden) an Prozess mit PID Signal 9: System beendet den Prozess mit PID	`kill` *pid* `kill -9` *pid*
`killall`	Signal 15 an alle Prozesse mit diesem Namen Signal 9 beendet alle Prozesse dieses Namens	`kill` *name* `kill -9` *name*
`nice`	unwichtiges Programm starten: kleine Priorität wichtiges Programm starten: höhere Priorität	`nice` *prg* `sudo nice -n -10` *prg*
`pgrep`	Liste der Prozesse mit String *str* im Namen	`pgrep` *str*
`pidof`	PID eines Pythonprogramms, Hintergrundprozess	`pidof python`
`ps`	Liste aller (`e`) Prozesse mit Einzelheiten (`f`)	`ps -ef`
`sync`	Speicherung beenden (bevor noch was passiert)	`sync`
`time`	Ausgabe der Zeit die ein Programm braucht	`time` *prg*
`timeout`	Beenden von Programm *prg* nach *n* Sekunden	`time` *n prg*
`top`	aktualisierte Liste der Prozesse des Nutzers `pi`	`top -u ahg`
`uptime`	aktuelle Zeit und Zeit seit Systemstart, Anzahl angemeldeter Nutzer	
`w`	1. Zeile wie `uptime`, dann aktueller Prozess für Nutzer in Terminals	
`watch`	Wiederholung alle `n` Sekunden, Abbruch: Ctrl-C	`watch -n1 date`

Rechnen

`bc`	Universalrechner, siehe Seite 224, Ende mit `quit`	`echo "3.5*2" | bc -l`
`calc`	(nach Installation) Rechner für sehr große Zahlen	`calc "2**9999 - 1"`
`dc`	Rechner mit umgek. poln. Notation, Ende mit `q`	`echo "6k 3.5 2 * p" | dc`
`expr`	Rechnen + - * / mit ganzen Zahlen, Leerzeichen!	`expr 7 \* 6`
`factor`	Primfaktorzerlegung natürlicher Zahlen	`factor 144`
`python`	Rechnen mit Python	`echo "import math; print(math.pi)" | python`

`qalc`	(nach Installation) guter Rechner mit Dezimalkomma	`qalc 3.5/2`
	Wechselkurse (`-e` update exch. rates) CNY EUR USD	`qalc -e 10 GBP`

4.1.3 Textausgabe, Datum und Zeit, Zeitsteuerung

Textausgabe

Der Standardbefehl zur Ausgabe von Zeichenketten (Text) ist `echo`. Mit `tput` kann man den Text formatieren (Farbe, Fettdruck), wobei aber die Möglichkeiten in der Konsole beschränkt sind. Für die Ausgabe von Textdateien auf dem Bildschirm oder in eine andere Datei eignet sich `cat`. Die genannten Befehle wollen wir uns genauer ansehen:

echo ist ein Befehl zur Ausgabe einer Zeichenkette oder eines Texts auf dem Bildschirm. Mithilfe einer Umleitung kann der Text stattdessen in eine Datei geschrieben werden. Den Befehl `echo` gibt es jedoch doppelt: a) unabhängig von der Shell als Teil der *coreutils* (GNU core utilities), aufgerufen zum Beispiel mit dem Befehl

```
/bin/echo "Hallo Welt!"
```

und b) als Befehl der Shell, hier: der Bash, der mit

```
echo "Hallo Welt!"
```

aufgerufen wird. Beide `echo`-Befehle haben folgende Optionen:

--help Ausgabe von Hilfe zum Befehl `echo`
-n ohne `newline` am Zeilenende (danach keine neue Zeile)
-E keine Auswertung von Escape-Sequenzen (die mit Backslash \beginnen), Standard
-e mit Auswertung von Escape-Sequenzen (die mit Backslash \beginnen), siehe unten

Wir verwenden in der Regel den `echo` Befehl der Shell. Welche Escape-Sequenzen ausgewertet werden, hängt vom Betriebssystem beziehungsweise der Bash-Version ab. Der `echo`-Befehl der *coreutils* gibt auf `/bin/echo --help` einen Hilfe-Text aus.

Mit dem Umlenkungsoperator `>` für die Ausgabe kann man in eine Datei schreiben:

```
echo "erste Zeile in Datei" > dat.txt
```

schreibt die Zeichenkette `erste Zeile in Datei` in die Datei `dat.txt`. Achtung: Wenn es `dat.txt` bereits gibt, wird der alte Inhalt überschrieben (das heißt: vor dem Schreiben der neuen Zeichenkette gelöscht). Wenn es die Datei `dat.txt` noch nicht gibt, wird sie automatisch erzeugt. Mit dem Umlenkungsoperator `>>` kann man Text anhängen:

```
echo "zweite Zeile in Datei" >> dat.txt
```

tput ist ein Befehl mit dem man in einem Terminal Text hervorheben (auszeichnen) kann. Man kann die Textfarbe (foreground color) und die Hintergrundfarbe (background color) ändern. Außerdem kann der Cursor (Eingabezeiger) bewegt werden. Der Befehl `tput` wird mit einer entsprechenden Option gegeben. `tput bold` lässt den Text fettgedruckt erscheinen und die Option `sgr0` stellt den normalen Zustand wieder her. Zum Beispiel ergibt

```
tput bold; tput setaf 6; echo "Hallo"; tput sgr0
```

ein fettes cyanblaues **Hallo**. Hier einige Optionen:

bold	Fettdruck
smul	Unterstreichung
setaf 0	Textfarbe Schwarz
setaf 1	Textfarbe Rot
setaf 2	Textfarbe Grün
setaf 3	Textfarbe Gelb
setaf 5	Textfarbe Magenta
setaf 6	Textfarbe Cyan
setab 4	Hintergrund Blau
setab 7	Hintergrund Weiß
sgr0	Normalzustand (Weiß auf Schwarz)

Der Befehl `tput` kann mit den Optionen `lines` beziehungsweise `cols` die Größe des Terminalbildschirms anzeigen, also die Zahl der Zeilen und Spalten.

```
echo "$(tput lines) Zeilen und $(tput cols) Spalten"
```

Um Bildschirmfläche zu säubern, gib man den Befehl

```
tput clear
```

ein. Der Cursor erscheint dann, nach dem Prompt in der ersten Zeile.

cat ist ein Befehl mit mehreren Anwendungen. In der Form

```
cat -n datei
```

dient er zur Anzeige einer kurzen Datei (hier `datei`) auf dem Bildschirm. Dabei erzeugt die Option `-n` Zeilennummern. Information zu weiteren Optionen gibt der Befehl

```
cat --help |  less
```

Durch Umleitung der Ausgabe von der Standardausgabe (Bildschirm) in eine andere Datei, kann man eine Datei kopieren. Der Befehl

```
cat datei1 > datei2
```

kopiert den Inhalt von Datei `datei1` in `datei2`. Existiert `datei2`, wird sie überschrieben, ansonsten wird sie erzeugt. Um eine kurze Textdatei zu schreiben, gibt man den Befehl

```
cat > datei
```

und schreibt dann zeilenweise einen Text, bis die Eingabe durch `Ctrl-d` beendet wird. `cat` hat seinen Namen von der Möglichkeit, die Inhalte zweier Dateien aneinander zu fügen. Diesen Vorgang nennt man Konkatenation (englisch con**cat**enation). Der Befehl

```
cat datei1 datei2 > datei3
```

schreibt die zusammengefügten Inhalte in eine dritte Datei. Die Datei `datei3` wird erzeugt, falls es sie nicht gibt. Gibt es sie bereits, wird sie geleert und dann die Inhalte von `datei1` und `datei2` eingefügt. Wenn die dritte Datei gleich der ersten ist, also auch `datei1`, enthält sie am Ende nur den Inhalt von `datei2`. Dagegen wird mit dem Befehl

```
cat datei2 >> datei1
```

der Inhalt von `datei2` der `datei1` angefügt.

Datum und Zeit

Der Befehl `date` gibt Datum und Zeit aus. Wenn die länderspezifischen Einstellungen bei der Konfiguration des Betriebssystems gesetzt wurden, erhält man die Namen von Monaten und Wochentagen in Deutsch. Die Ausgabe wird durch eine Zeichenkette bestimmt, die mit einem Pluszeichen + beginnt (das nicht ausgegeben wird) und unter anderem folgende Zeichenpaare mit besonderer Bedeutung enthalten kann.

%A der Wochentag, zum Beispiel Montag
%a der Wochentag in dreibuchstabiger Abkürzung
%B der Monatsname, zum Beispiel Januar
%b der Monatsname in dreibuchstabiger Abkürzung
%d der Tag des Monats (eine Zahl zwischen 1 und 31)
%H die Stunde des Tages (eine Zahl zwischen 0 und 23)
%h die Stunde des (halben) Tages (eine Zahl zwischen 1 und 12)
%j der Tag im aktuellen Jahr (eine Zahl zwischen 1 und 366)
%m der Monat als Zahl (eine Zahl zwischen 1 und 12)
%M die Minute (eine Zahl zwischen 0 und 59)
%n Einfügung einer neuen Zeile
%S Sekunde (eine Zahl zwischen 0 und 59)
%u der Tag der Woche (eine Zahl zwischen 1 und 7, Montag = 1)
%V Kalenderwoche (gemäß ISO 8601)
%X Zeit (Stunde:Minute:Sekunde)
%x Datum (Tag.Monat.Jahr)
%Y Jahr
%y Jahr, abgekürzt auf zwei Ziffern
%Z Zeitzone (CET = Central European Time = Mitteleuropäische Zeit)

%% das Prozentzeichen %

Bei Zahlen, die ein- oder zweistellig sein können, legt man mit einem der folgenden Zeichen, *flag* genannt, fest, wie sie dargestellt werden sollen, falls sie einstellig sind.

- einstellige Ausgabe (zum Beispiel 1.1.2020)
_ Leerzeichen vor der Ziffer (zum Beispiel 1. 1.2020)
0 Null vor der Ziffer (zum Beispiel 01.01.2020), Standard (ohne *flag*)

Die *flag* folgt auf das Prozentzeichen. Man erhält ein Zeichentripel. Zum Beispiel ergibt

```
date "+Zeit: %_d.%_m. %Y %0H:%0M:%0S"
```

am 2. März des Jahres 2020 die Ausgabe `Zeit: 2. 3. 2020 00:10:08`.

Weiterhin gibt es folgende Befehle für Datum und Zeit:

`calcurse` empfehlenswertes Programm zur Terminplanung, siehe Seite 231

`calendar` Programm zur Terminplanung, besser ist `calcurse` oder `when`

`ncal` (erfordert Installation des Pakets `bsdmainutils`) zeigt Kalendermonat, `-b` zeilenweise, `-y` ganzes Jahr, Beispiel: `ncal -b -3` drei Monate in einer Zeile

`sleep` Programm wartet *x*`m` Minuten und *x*`s` Sekunden Beispiel: `time 10s`

`when` Kalender und Terminplanung, siehe Seite 233

Zeitsteuerung mit Cron

Programme, die periodisch zu bestimmten Zeiten ausgeführt werden sollen, zum Beispiel zu jeder vollen Stunde, können als sogenannte *Cronjobs* (von altgriechisch *χρόνος*, die Zeit) automatisch vom *Cron-Dämon* gestartet werden. Das ist ein Betriebssystemprogramm, welches in der Regel standardmäßig im Hintergrund läuft. In Betriebssystemen, die mit systemd arbeiten, wie Raspberry Pi OS, kann man den Zustand des Cron-Dämons mit dem Befehl

```
systemctl status cron
```

erfragen und in der Antwort sollte `active (running)` stehen.

Nachteilig ist, dass die Programme nicht ausgeführt werden, wenn der Rechner zu den bestimmten Zeiten ausgeschaltet ist. (Dies Problem kann mit dem Programm `anacron` gelöst werden.) Man kann ein Programm mit `cron` auch nicht zu einem beliebigen Zeitpunkt einmalig ausführen lassen. Allerdings kann man den Cron-Dämon anweisen, Programme nach dem Starten des Rechners auszuführen.

Die Information zu Cronjobs wird für jeden Nutzer (einschließlich root) in einer eigenen Datei, Crontab genannt, gespeichert. Crontabs der Nutzer stehen im Verzeichnis `/var/spool/cron/crontabs`. Außerdem gibt es eine sytemweite Crontab namens `crontab` im Verzeichnis `/etc`.

Um die Arbeit des Cron-Dämons nicht zu stören, wird die Crontab eines Nutzers folgendermaßen bearbeitet. Im Terminalfenster erlaubt der Befehl

```
EDITOR=nano crontab -e
```

die Änderung (*e* für edit) der Crontab des aktuellen Nutzers mit dem Editor `nano`. Statt `nano` kann auch ein anderer Editor gewählt werden. Falls noch keine Crontab existiert, wird sie angelegt. In der Crontab wird jeder Cronjob in einer Zeile beschrieben, in der eine Zeitangabe und das auszuführende Programm stehen. Am Ende der Crontab muss eine Leerzeile oder Kommentarzeile stehen. Für die auszuführenden Programme müssen die Rechte so gesetzt sein, dass sie vom Nutzer ausgeführt werden können.

Die Crontab eines Nutzers hat sechs Spalten, die durch Leerzeichen oder Tabulatoren getrennt werden. Die Zeitangabe besteht normalerweise aus den ersten fünf Spalten mit Sternchen $*$ oder Zahlen: Minute (0 – 59), Stunde (0 – 23), Tag des Monats (1 – 31), Monat (1 – 12), Tag der Woche (0 – 6, 0 ist Sonntag). Spalten, die nicht zur Einschränkung der Zeit gebraucht werden, erhalten ein Sternchen. Zum Beispiel wird mit

```
11 11 11 11 * /home/ahg/shell/helau.sh
```

die Shell-Skripte `helau.sh` im Verzeichnis `/home/ahg/shell` jedes Jahr am 11. 11. um 11 Uhr 11 ausgeführt.

Mehrere Werte in einer Spalte werden durch Komma (ohne folgendes Leerzeichen) getrennt. Das Zeichen - gibt Bereiche an und das Zeichenpaar $*/$ Zeitintervalle. Mit

```
*/5 8-10 * * 2,3 /home/ahg/shell/messung.sh
```

wird Dienstags und Mittwochs zwischen 8 und 10 Uhr alle 5 Minuten die Shell-Skripte `messung.sh` im Verzeichnis `/home/ahg/shell` ausgeführt.

Manche Zeitangaben haben eine kürzere Schreibweise (ersetzt die ersten fünf Spalten):

`@reboot`	nach jedem Start des Betriebssystems	
`@weekly`	wöchentlich, entspricht	`0 0 * * 0`
`@daily`	täglich, entspricht	`0 0 * * *`
`@hourly`	stündlich, entspricht	`0 * * * *`

Am Ende der Crontab muss eine Leerzeile stehen.

Cronjobs werden im Hintergrund (background) gestartet. In der entsprechenden Zeile der Crontab-Datei ist darum der Zusatz & nach dem Programmnamen überflüssig. Ausgaben eines Cronjobs werden standardmäßig als E-Mail an den Nutzer gesandt, dessen Crontab den Cronjob gestartet hat (und nicht auf dem Bildschirm gezeigt). Man kann den Standardausgabestrom (`stdout`, Datei-Deskriptor: 1) jedoch in eine Datei umleiten. Wählt man `/dev/null` als Datei, wird die Ausgabe verworfen. Der Standardfehlerausgabestrom `stderr` hat den Datei-Deskriptor 2. Mit der Zeile

```
@reboot python3 /home/ahg/progs/mytest.py >/dev/null 2>&1
```

oder, wenn das Pythonprogramm ausführbar ist und eine Shebang-Zeile enthält (siehe unten, Seite 261), kürzer mit

```
@reboot /home/ahg/progs/mytest.py >/dev/null 2>&1
```

wird das Pythonprogramm `mytest.py`, das im Verzeichnis `/home/ahg/progs` steht, nach jedem Systemstart ausgeführt und die Ausgabe, einschließlich Fehlermeldungen (darum der Zusatz `2>&1`) verworfen. Auf Seite 380 findet man ein Beispiel, in dem mithilfe eines Cronjobs ein Display minütlich aktualisiert wird und die Zeit anzeigt.

Mit dem Befehl

```
crontab -l
```

wird die Crontab des aktuellen Nutzers angezeigt (l für list) und mit

```
crontab -r
```

wird die Crontab des aktuellen Nutzers gelöscht (r für remove).

Zeitsteuerung mit systemd

Wenn das Betriebssystem mit systemd arbeitet (siehe Seite 12), wie Raspberry Pi OS, kann man `systemd` (statt des Cron-Dämons) zur Zeitsteuerung von Prozessen (laufenden Programmen) benutzen. Das Programm, welches gestartet werden soll, wird von `systemd` in einer *service unit* beschrieben und die Zeitsteuerung dazu in einer *timer unit*.

Dafür erstellt man zwei Konfigurationsdateien (Textdateien, die in `systemd` als *unit files* bezeichnet werden), deren Namen mit einem wählbaren Anfangsteil beginnen (zum Beispiel mit `meindienst`) und auf `.timer` beziehungsweise `.service` enden. Beide Dateien sollen im Verzeichnis `/etc/systemd/system` stehen und können mit einem Texteditor wie `nano` (siehe Seite 131) erzeugt werden. Man braucht Administratorrechte und daher wird zum Schreiben von `meindienst.service` der Befehl

```
sudo nano /etc/systemd/system/meindienst.service
```

gegeben, und entsprechend für `meindienst.timer`. Im folgenden Beispiel soll ein Pythonprogramm `dienst.py`, das im Verzeichnis `/home(ahg/progs/` steht, einmal pro Minute aufgerufen werden. Das Pythonprogramm `dienst.py`

```
1 # Programm dienst.py
2 print("Hallo Welt")
```

schreibt die Zeichenkette `Hallo Welt` auf die Standardausgabe (auf den Bildschirm, wenn es keine Ausgabeumlenkung gibt). Die Datei `meindienst.timer` erhält den Code

```
# meindienst.timer (AHG, June 2022-06-03)

[Unit]
Description=start_service_meindienst.service
Requires=meindienst.service

[Timer]
```

```
Unit=meindienst.service
OnCalendar=minutely
AccuracySec=10seconds

[Install]
WantedBy=timers.target
```

Ein unit file vom Typ `timer` muss einen Abschnitt enthalten, der mit `[Timer]` überschrieben ist. Der Wert von `Unit` gibt an, welche unit nach der Auslösung der Zeitsteuerung aktiviert werden soll. Standardwert ist eine unit, deren Name mit dem gleichen Anfangsteil beginnt wie die timer unit, und die auf `.service` endet.

Der Wert von `OnCalendar` gibt an, wann der Timer eine Aktion auslösen soll. Der Wert `minutely` ist eine Kurzform von `*-*-* *:*:00` und steht für jede volle Minute. Eine Übersicht über Kurzformen für Werte von `OnCalendar` findet sich in Tabelle 4.2.

Zeit	Kurzform des Werts	Langform des Werts
am Anfang jeder vollen Minute	minutely	`*-*-* *:*:00`
am Anfang jeder vollen Stunde	hourly	`*-*-* *:00:00`
an jedem Tag um 0 Uhr	daily	`*-*-* 00:00:00`
montags um 0 Uhr	weekly	`Mon *-*-* 00:00:00`
am ersten Monatstag um 0 Uhr	monthly	`*-*-01 00:00:00`
an jedem Jahresanfang	yearly	`*-01-01 00:00:00`
Montag bis Freitag um 8 Uhr 29	`Mon..Fri 08:29`	`Mon..Fri *-*-* 08:29:00`
alle 10 Minuten	`*:0/10`	`*-*-* *:0/10:00`

Tabelle 4.2: Einige Werte von `OnCalendar` für eine timer unit von `systemd`

Die Auslösung darf aber auch ein bisschen später erfolgen (standardmäßig bis zu einer Minute später). Mit der Zeile `AccuracySec=10seconds` verlangen wir, dass die Aktion spätestens 10 Sekunden nach der vollen Minute ausgelöst werden muss.

Es dürfen mehrere `OnCalendar`-Zeilen eingesetzt werden. Die Auslösung der Zeitsteuerung erfolgt dann zu allen angegebenen Zeitpunkten. Die Auslösung kann auch durch *monotonic timers* gesteuert werden, wobei der Auslösezeitpunkt nach einem bestimmten Ereignis festgelegt wird, insbesondere nach dem Booten.

Um zu prüfen, ob die Datei `/etc/systemd/system/meindienst.timer` den Regeln von `systemd` entspricht, geben wir den Befehl

```
systemd-analyze verify /etc/systemd/system/meindienst.timer
```

ein. Wenn es keine Fehlermeldung gibt, wurde die Datei erfolgreich geprüft (verifiziert). Die Datei `meindienst.service` erhält den Code

```
# meindienst.service (AHG, June 2022-06-03)

[Unit]
```

```
Description=start_pythonprogram_for_user_ahg
Wants=meindienst.timer

[Service]
Type=simple
ExecStart=/usr/bin/python3 /home/ahg/progs/dienst.py
User=ahg
StandardOutput=tty
StandardError=tty

[Install]
WantedBy=multi-user.target
```

Der Wert von `ExecStart` gibt an, welcher Befehl von diesem Dienst ausgeführt werden soll. `/usr/bin/python3` ist der Pfad zum Python interpreter, den wir mit dem Befehl `which python3` erfahren. In der Zeile `User=ahg` legen wir fest, dass der Dienst unter dem Nutzer `ahg` laufen soll (Standard ist `root`). Die Ausgabe des mithilfe von `ExecStart` aufgerufenen Programms (hier: `dienst.py`) wird normalerweise nicht in die Standardausgabe oder Standardfehlerausgabe geschrieben, sondern in eine besondere Datei (systemd journal), siehe unten. Die Zeilen `StandardOutput=tty` und `StandardError=tty` bewirken, dass die Ausgabe stattdessen auf dem Bildschirm (`tty`) erfolgt.

Um zu prüfen, ob die Datei `/etc/systemd/system/meindienst.service` den Regeln von `systemd` entspricht, geben wir den Befehl

```
systemd-analyze verify /etc/systemd/system/meindienst.service
```

ein. Ohne Fehlermeldung wurde die Datei verifiziert. Danach starten wir `systemd` erneut

```
sudo systemctl daemon-reload
```

und dann setzen wir den automatischen Start des Dienstes nach dem Booten:

```
sudo systemctl enable meindienst.timer
```

Schließlich starten wir den Dienst mit dem Befehl

```
sudo systemctl start meindienst.timer
```

und nun wird zu Beginn jeder vollen Minute die Zeichenkette `Hallo welt` auf den Bildschirm geschrieben. Da das Programm `dienst.py` im Hintergrund läuft, kann man weiterhin mit anderen Programmen in der Konsole arbeiten. Allerdings wird man minütlich durch ein `Hallo welt` gestört.

Um den automatischen Start des Dienstes nach dem nächsten Start des des Betriebssystems (Booten) auszuschalten, geben wir den Befehl

```
sudo systemctl disable meindienst.timer
```

In der aktuellen Sitzung läuft der Dienst aber noch weiter. Um den Dienst jetzt abzustellen, gibt man den Befehl

```
sudo systemctl stop meindienst.timer
```

ein. Statt des obigen, ziemlich sinnlosen Pythonprogramms `dienst.py` kann man natürlich ein anderes nehmen und zum Beispiel auf einem externen Display die Zeit anzeigen, siehe das Programm `lcd_clock.py` auf Seite 380 (vorausgesetzt, dass ein Display angeschlossen wurde).

Jedes Mal, wenn eine service unit startet oder beendet wird, schreibt `systemd` eine Zeile in die zugehörige Protokolldatei, welche normalerweise auf der Speicherkarte liegt. Will man die Speicherkarte schonen, kann man die gesamte Protokollierung durch `systemd` im RAM ablegen. Dann verschwinden die neuen Protokolleinträge beim Ausschalten des Rechners. Dies wird erreicht, indem mit Administratorrechten die Konfigurationsdatei

`/etc/systemd/journald.conf`

verändert wird: Die Zeile `#Storage=auto` wird durch `Storage=volatile` ersetzt.

Wenn die die Zeilen `StandardOutput=tty` und `StandardError=tty` in der Datei der service unit (`meindienst.service`) fehlen, wird die Ausgabe des mithilfe von `ExecStart` aufgerufenen Programms (hier: `dienst.py`) von `systemd` protokolliert, aber nicht auf dem Bildschirm ausgegeben. Die Ausgabe von `dienst.py` kann man lesen, wenn die Protokolldatei für die service unit `dienst.service` geöffnet wird:

```
journalctl -u meindienst.service --since today
```

Dabei bewirkt die Option `-since today` die Beschränkung auf das heutige Protokoll. Statt `meindienst.service` genügt in obigem Befehl schon `meindienst`. Falls man das Protokoll für die timer unit anschauen will, gibt man entsprechend den Befehl

```
journalctl -u meindienst.timer --since today
```

aber das ist meistens nicht von Interesse, außer bei der Fehlersuche.

Mit den beiden Befehlen (die nacheinander gegeben werden)

```
sudo journalctl --rotate
sudo journalctl --vacuum-time=1s
```

kann man die Protokolldateien (journal files) von `systemd` leeren, was jedoch meistens nicht notwendig ist. Neue Protokolldateien werden automatisch angelegt.

4.1.4 Information vom Betriebssystem

Befehle

Der Befehl

```
hostnamectl
```

gibt den Namen des Rechners (network node hostname), eine Nummer, die bei der Installation des Betriebssystems vergeben wurde (machine ID), eine Nummer, die beim Start (boot) des Rechners erzeugt wurde (boot ID), den Namen des Betriebssystems (operating system), die Version (Release) des Linux-Kernels (kernel release), und den Prozessortyp (architecture) aus. Der Rechnername (hostname) oder die machine ID dienen dazu, den Rechner in einem Netzwerk zu identifizieren. Die machine ID steht in der Datei `/etc/machine-id`; sie sollte nicht öffentlich bekannt gemacht werden. Die boot ID steht in der Datei `/proc/sys/kernel/random/boot_id`; sie ändert sich beim Neustart des Rechners.

Auch der Befehl `uname` gibt Information des Linux-Kernels aus. Mit den Optionen `-m`, `-n` oder `-r` erhält man den Prozessortyp (machine hardware name) oder den Namen des Rechners oder die Version (release) des Kernels.

Mit dem Befehl

```
uptime -p
```

erfährt man, wie lange das Betriebssystem in der aktuellen Sitzung schon läuft. Die Option `-p` ergibt ein schöneres Format (p für pretty) und beschränkt die Ausgabe auf die genannte Dauer ohne weitere Information.

Neben dem Betriebssystem (Raspbian) benötigt man zum Betrieb des Rechners die Firmware. Sie wird wird von der Firma Broadcom bereitgestellt und ist keine freie Software. Sie befindet sich in der ersten Partition der Speicherkarte. Mit

```
vcgencmd version
```

kann man sich die Version der Firmware anzeigen lassen. Das Programm `vcgencmd` kann mit verschiedenen anderen Optionen aufgerufen werden.

```
vcgencmd measure_temp
```

gibt die Temperatur der GPU aus und

```
vcgencmd measure_volts
```

gibt die Spannung der GPU aus (Soll: 1,2 V). Mit

```
vcgencmd codec_enabled H264
```

erfährt man ob der videoodec (siehe Seite 66) H264 installiert ist. Da H264 mitgeliefert wird, ist die Antwort `enabled`. Dagegen muss der Codec MPG2 gekauft und installiert werden, sonst ist er `disabled`. Codecs, deren Installation entsprechend mit obigem Befehl erfragt werden kann, sind H264, MPG2, WVC1, MPG4, MJPG und WMV9.

Der Arbeitsspeicher für CPU und GPU wird mit den Befehlen

```
vcgencmd get_mem arm
```

beziehungsweise

```
vcgencmd get_mem gpu
```

ausgegeben. Die Taktrate der CPU wird nach

```
vcgencmd measure_clock arm
```

angezeigt.

Den Namen des angemeldeten Nutzer kann man mit dem Befehl

```
logname
```

erfahren. Dagegen wird der aktuelle Nutzer mit dem Befehl

```
whoami
```

angezeigt. Wird `whoami` in einem Terminal gegeben, ist der aktuelle Nutzer gleich dem angemeldeten. Aber ein Programm, zum Beispiel ein Shellscript, kann auch von einem anderen Nutzer (zum Beispiel Root) ausgeführt werden. Innerhalb des Scripts ergibt `whoami` dann einen aktuellen Nutzer, der nicht der angemeldete Nutzer ist. Die Belegung der Speicherkarte erfährt man mit dem Befehl

```
df -h
```

in der Zeile mit `dev/root`. Der belegte Speicher wird auch in Prozent angegeben.

Eine kurze Beschreibung der Schnittstellen des Raspberry PI, einschließlich der Belegung der GPIO, erhält man mit dem Befehl

```
pinout | less
```

Umgebungsvariablen

Umgebungsvariablen sind globale Variablen des Betriebssystems, die Information für die Shell und andere Programme speichern. Die Ausführung von Betriebssystembefehlen und Shell-Skripten wird oft durch den Wert von Umgebungsvariablen beeinflusst. Umgebungsvariablen werden automatisch beim Start des Betriebssystems auf einen bestimmten Wert gesetzt. Dieser kann, unter anderem mit dem Befehl `set` der Bash, verändert werden. Wichtige Umgebungsvariablen sind:

BASH Pfad zur aktuellen Shell, bei Raspbian in der Regel: `\bin\bash`

BASH_ENV Name einer Datei, die (statt `.bashrc`) beim Shell-Start ausgeführt wird

BASH_VERSION Version der aktuellen Shell (`4.3.30(1)-release` oder höher)

EUID effektiver Nutzer-ID des aktuellen Benutzers (`1000` nach normalem Log-in)

HOME Home-Verzeichnis des aktuellen Benutzers (`\home\pi` nach normalem Log-in)

HOSTNAME Rechnername (`raspberrypi`, wenn kein neuer Name gewählt wurde)

IFS Internal Field Separator, Trennzeichen: beliebig viele Leerzeichen, Tabs, Newlines

LANG länderspezifischer Zeichensatz (`de_DE.UTF-8` nach deutscher Konfiguration)

LINENO aktuelle Zeilennummer (kann beim Debugging von Skripten nützlich sein)

LOGNAME Login-Name des aktuellen Benutzers (`ahg` nach normalem Log-in)

MAIL Mailbox, Name einer Datei zur Speicherung ankommender E-Mails

MAILCHECK Periodendauer (in s) der Prüfung von E-Mail-Ankunft (Vorgabe: 60)

PATH Liste von Pfadnamen, durch : getrennt, für die Suche nach Kommandos

PPID Prozess-ID des Vater-Prozesses (Subshell = Shell-Kopie ⇒ behält diese PPID)

PS3 Eingabe-Prompt von Menüs, die mit einem Shell-Skript erzeugt wurden

PWD Pfad zum aktuellen Arbeitsverzeichnis

RANDOM ganze Zufallszahl z, $0 \leq z \leq 32767$; Initialisierung mit `RANDOM=n`

REPLY in einem Skript: gewählte Nummer in Menü oder eingelesene Zeile bei `read`

SECONDS Laufzeit (in s) der aktuellen Shell, kann auf neuen Wert gesetzt werden

SHLVL shell level, Verschachtelungstiefe der Shell (`1` für eine Konsolen-Shell)

TERM Terminal-Typ (Bildschirmsteuersequenzen werden in einer Datei beschrieben)

UID Nutzer-ID des aktuellen Benutzers (`1000` nach normalem Log-in)

Den Wert einer Umgebungsvariablen `X` erfahren wir im Terminal mit dem Befehl

```
echo $X
```

Einige Umgebungsvariablen haben bereits einen festen Wert, bevor wir sie abfragen, anderen wird erst während der Abfrage ein Wert zugewiesen, zum Beispiel `SECONDS`. Eine Liste der Umgebungsavariablen mit festem Wert erhält man mit

```
printenv
```

4.1.5 Ansteuerung der GPIO

GPIO sysfs Interface (veraltet)

Für den Zugriff auf Geräte und Betriebssystemteile stellt der Linux Kernel ein virtuelles Dateisystem namens *sysfs* zur Verfügung. Damit ist es auch möglich GPIO zu steuern. Dies ist jedoch nicht zu empfehlen und gilt als veraltet (deprecated). Da jedoch die Ansteuerung der GPIO über sysfs in der Literatur zum Raspberry Pi oft genannt wird, betrachten wir ein Beispiel. Später behandeln wir bessere Möglichkeiten.

Wir gehen in unserem Beispiel davon aus, dass eine Leuchtdiode (LED) an GPIO 23 angeschlossen ist, siehe Abschnitt 1.2.4 auf Seite 32. Man kann auch die auf Seite 34 beschriebene Transistorschaltung verwenden.

Wir wechseln in das virtuelle Verzeichnis `gpio`

```
cd /sys/class/gpio
```

und schreiben die Nummer des GPIO in die virtuelle Datei `export`[2]

```
echo 23 > export
```

Der Befehl `echo` schreibt die nachfolgende Zeichenkette (hier `23`) auf den Monitor (Standardausgabe), wenn keine Umleitung erfolgt. In obigem Befehl wird aber mit dem „größer als“-Zeichen `>` die Ausgabe umgeleitet und in die Datei `export` geschrieben. Die Zeichenkette `23` enthält keine Leerzeichen und kann, muss aber nicht, in einfachen `'` oder doppelten Anführungszeichen `"` eingefasst werden. Automatisch wird ein Zeilenumbruch angefügt (sofern er nicht mit der Option `-n` unterdrückt wird).

Wir wechseln in das virtuelle Verzeichnis `gpio23`, das automatisch angelegt wurde),

```
cd gpio23
```

und schreiben `out` in die virtuelle Datei namens `direction`

```
echo out > direction
```

und `1` in die virtuelle Datei `value`

```
echo 1 > value
```

Die LED sollte nun leuchten. Um sie auszuschalten, geben wir schließlich den Befehl

```
echo 0 > value
```

Damit GPIO 23 wieder von anderen Programmen angesprochen werden kann, gehen wir in das übergeordnete virtuelle Verzeichnis `/sys/class/gpio` zurück

```
cd ..
```

und geben GPIO 23 frei mit dem Befehl

```
echo 23 > /sys/class/gpio/unexport
```

pigpio, der Dämon pigpiod und das Programm pigs

`pigpio` ist eine, in der Programmiersprache C geschriebene Bibliothek zur Ansteuerung der GPIO. Sie wird mit dem Paket `pigpio` installiert. Es ist allerdings nicht sicher, ob `pigpio` auch mit neuen Modellen des Raspberry Pi funktioniert. Vielleicht benötigt man eine neuere Version von `pigpio`.

Andere Programmiersprachen können die Bibliothek benutzen, wenn der Daemon `pigpiod` läuft. Er wird durch

[2] `echo` schreibt die nachfolgende Zeichenkette (hier `23`) auf den Monitor (Standardausgabe), wenn keine Umleitung erfolgt. Die Zeichenkette kann auch in einfachen oder doppelten Anführungszeichen eingefasst werden. In obigem Befehl wird aber mit dem Zeichen `>` die Ausgabe umgeleitet und in die virtuelle Datei `export` geschrieben. Obwohl man, aus Sicht des Nutzers, in die virtuelle Datei `export` schreiben kann, sind andere Dateioperationen, zum Beispiel Lesen, nicht möglich.

```
sudo pigpiod
```

gestartet. Nach dem Start von `pigpiod` ermöglicht das Programm `pigs` die Ansteuerung der GPIO von der Kommandozeile oder in einem Shellscript.

Wir gehen zunächst davon aus, dass eine Leuchtdiode (LED) an GPIO 23 angeschlossen ist, wie in Abschnitt 1.2.4 auf Seite 32 beziehungsweise Seite 34 beschrieben.

Zum Einschalten der LED genügt nun der Befehl

```
pigs w 23 1
```

`w` ist die Kurzform von `WRITE` (Schreiben). `23` bezeichnet GPIO 23 und `1` den logischen Zustand „high“, entsprechend 3,3 V. Zum Ausschalten der LED genügt

```
pigs w 23 0
```

wobei `0` den logischen Zustand „low“ bezeichnet, entsprechend 0 V.

Zur Abfrage des logischen Zustands eines GPIO (hier `23`) dient der Befehl

```
pigs r 23
```

`r` ist die Kurzform von `READ` (Lesen). Der Befehl gibt `0` oder `1` zurück. `0` bezeichnet wieder den logischen Zustand „low“ und `1` den Zustand „high“.

Wir wollen nun die Helligkeit der LED vom Programm verändern lassen (Dimmen). Dies kann nicht durch Veränderung des Vorwärtsstroms der Diode geschehen, da der GPIO eine feste Ausgangsspannung von 3,3 V liefert und der Wert des Vorwiderstands sich nicht vom Programm verändern lässt. Die mittlere Helligkeit der LED kann man aber verringern, indem man sie abwechselnd ein- und ausschaltet. Das Schalten soll nun so schnell erfolgen, dass unser Auge es nicht bemerkt. Diese Methode heißt Dimmen durch Pulsweitenmodulation. `pigpio` bietet Pulsweitenmodulation für GPIO 1 bis 31 an, wobei die Zeitintervalle für das Ein- und Ausschalten durch eine interne Uhr der Elektronik geliefert werden (hardware timing). Mit dem Befehl

```
pigs p 23 100
```

wird GPIO 23 auf den Wert 100 gedimmt, auf einem Wertebereich von 0 (ganz aus) bis 255 (ganz an). `p` ist die Kurzform von `PWM`.

Die LED wird beim Dimmen mit einem pulsbreitenmodulierten periodischen Rechteck-Signal betrieben. Das Signal besteht aus einem Wechsel von logischer 0 (0 V) und logischer 1 (3,3 V) mit Zeitdauern von t_{aus} beziehungsweise t_{ein}. Die Periodendauer des Signals ist $T = t_{\text{ein}} + t_{\text{aus}}$ und die Grundfrequenz ist $f = 1/T$. Der Tastgrad (duty cycle) ist $t_{\text{ein}}/(t_{\text{ein}}+t_{\text{aus}})$. Man kann die Signalform betrachten, indem der Zeitverlauf des Spannungsabfalls am Vorwiderstand der LED auf einem Oszilloskop sichtbar gemacht wird.

Wir gehen nun davon aus, dass ein Schalter an GPIO 22 angeschlossen ist, wie in Abschnitt 1.2.4 auf Seite 35 beschrieben. Die Abfrage des logischen Zustands des GPIO (hier `22`) geschieht, wie im oben (auf Seite 130) beschrieben, mit dem Befehl

```
pigs r 22
```

Folgendes Script startet den Daemon `pigpiod` und gibt fortlaufend die Schalterstellung auf dem Bildschirm aus. Drücken einer beliebigen Taste, zum Beispiel der Leertaste (ohne anschließendes ENTER), schaltet `pigpiod` aus und beendet das Programm.

```
#!/bin/bash
ende()
{
    sudo killall pigpiod
    echo -e "\n\npigpiod ausgeschaltet: Ende.\n"
    exit
}
sudo pigpiod
echo -e "\n\npigpiod eingeschaltet: Start"
echo -e "Abbruch durch Drücken einer Taste\n"
while true
do
    if read -n 1 -t 0.25 x ; then ende ; fi
    s=`pigs r 22` ; echo $s
done
```

4.2 Editoren

Ein Texteditor für das Terminal ist ein wichtiges und häufig gebrauchtes Werkzeug der Systemadministration und Programmierung. Man sollte sich für einen entscheiden und diesen gut beherrschen.

Wir stellen *nano* (leicht erlernbar) und *vi / vim* (gewöhnungsbedürftig) vor. GNU *Emacs* ist ein weiterer leistungsfähiger Texteditor, der mit dem Paket *emacs* installiert werden kann. Er ist sehr umfangreich und wir verzichten hier auf eine Beschreibung.

4.2.1 nano

nano ist ein einfacher Texteditor für das Terminal. Eine Dokumentation findet man im Verzeichnis `/usr/share/doc/nano`. Für eine Anleitung öffnet man die Datei `nano.html` mit einem Browser, zum Beispiel `w3m` (siehe Seite 209).

```
w3m /usr/share/doc/nano/nano.html
```

Mit `q` kann man den Browser `w3m` verlassen. Mehr Information findet man entsprechend in der Datei `faq.html`.

Der Befehl

```
nano -Y sh datei
```

öffnet zum Beispiel die Datei `datei`, die ein Shellscript enthält. Die Option `-Y sh` bewirkt Syntaxhervorhebung (Syntax-Highlighting). Wenn `nano` den Dateityp anhand der

Endung des Dateinamens (beispielsweise `.sh`) erkennt, wendet es automatisch Syntaxhervorhebung an. Mit der Option `-Y` kann man Syntaxhervorhebung für Dateien ohne entsprechende Endung bewirken.

Um zu erfahren, für welche Programmiersprachen und Scripte `nano` Syntaxhervorhebung zur Verfügung stellt, kann man sich mit

```
ls /usr/share/nano/
```

die Liste der entsprechenden Konfigurationsdateien ansehen.

Optionen

--help ohne weitere Optionen und ohne Dateiname: Hilfe (Liste der Optionen)
-l Zeilennummern (line numbers) anzeigen
-m Positionierung mit Maus möglich, wenn `gpm` installiert ist (siehe Seite 48)
-Y sh Syntaxhervorhebung für ein Shellscript ohne Endung `.sh`

Tastenbefehle

`Ctrl-x` bedeutet, dass die Tasten `Ctrl` und `x` gleichzeitig gedrückt werden. Auf deutschen Tastaturen kann die Control-Taste mit `Strg` beschriftet sein. Entsprechend bedeutet `Alt-x`, dass die Tasten `Alt` und `x` gleichzeitig gedrückt werden. `Shift-`→ ist gleichzeitiges Drücken der Taste `Shift` (Umschalttaste) und der Pfeiltaste →.

→ Cursor nach rechts bewegen (entsprechend ←, ↑ und ↓)
Shift-→ Zeichen rechts vom Cursor markieren (entsprechend ←, ↑ und ↓)
Ctrl-k markierte Zeichenkette ausschneiden (cut) und in Zwischenablage speichern
Ctrl-u den Inhalt der Zwischenablage hier einfügen (uncut)
Ctrl-k Ctrl-u markierte Zeichenkette in die Zwischenablage kopieren
Ctrl-w Suche nach einer Zeichenkette und Positionierung des Cursors
Ctrl-o den ganzen Text in der geöffneten Datei speichern
Ctrl-x die Datei, deren Textinhalt gezeigt wird, schließen
Alt-U Undo (Rückgängig) = letzte Editieroperation rückgängig machen
Alt-E Redo (Wiederholen) = voriges Undo aufheben

Copy & Paste aus anderer Datei

Oft möchte man bei der Bearbeitung einer Textdatei `datei` mit *nano* Textabschnitte aus einer anderen Textdatei `quelle` einfügen. Das kann man so machen:

1) Öffnung von Textdatei `datei` in *nano* mit Option `F`

   ```
   nano -F datei
   ```

2) Öffnung von Textdatei `quelle` im gleichen Editor

   ```
   Ctrl-r quelle
   ```

3) Markierung des Textabschnitts mit dem Cursor

4) Übertragung des Textabschnitts in die Zwischenablage
   `Ctrl-k Ctrl-u`

4) Schließen der Textdatei `quelle`
   `Ctrl-x` (gefolgt von `N`, um nicht zu speichern)

5) Positionierung des Cursors in Textdatei `datei`

6) Einfügen aus der Zwischenablage
   `Ctrl-u`

Will man mehrere Textabschnitte übertragen, kann man mit `Alt-<` (gleichzeitiges Drücken der Tasten `Alt` und `<`) zwischen den beiden Textdateien hin und her wechseln.

Konfiguration von nano

Das Verhalten des Editors `nano` kann durch eine Konfigurationsdatei im Heimatverzeichnis des Nutzers, namens `.nanorc`, angepasst werden, welche durch den Befehl

`nano ~/.nanorc`

geöffnet wird. Um immer Zeilennummern anzeigen zu lassen, fügt man die Zeile

`set linenumbers`

ein und speichert und schließt die Datei, wie oben bei den Tastenbefehlen beschrieben.

Es kann passieren, dass die Bearbeitung einer Datei misslingt und eine unbrauchbare Datei gespeichert wird. Dann ist es gut, wenn man auf die vorige Version zurückgreifen kann. Um dies zu erreichen, fügt man zusätzlich in die Konfigurationsdatei die Zeile

`set backup`

ein. Die alte Version der Datei wird dann automatisch gespeichert. Sie erhält den gleichen Namen wie die neue (bearbeitete) Datei, aber gefolgt vom Tildezeichen ~.

4.2.2 vi und vim

vi ist ein alter Editor (aus dem Jahr 1976) zur Bearbeitung von Textdateien in einem Terminal, welcher auf allen Betriebssystemen, die von Unix abstammen, verfügbar ist oder installiert werden kann. Eine Erweiterung ist *vim* (*vi* *im*proved), mit dem auf fast allen Betriebssystemen (einschließlich Windows-Versionen, Mac OS X, AmigaOS und OS/2) gearbeitet werden kann. Falls *vim* noch nicht installiert ist, erfolgt dies in üblicher Weise mit `sudo apt install vim`. Er braucht wenig Speicher und arbeitet sehr schnell, seine Bedienung ist jedoch ungewöhnlich. Wir betrachten hier hauptsächlich einige Grundfunktionen, die in *vi* und *vim* verfügbar sind, sowie einige Erweiterungen, die es nur in *vim* gibt.

Will man eine Datei *dat* mithilfe von nur lesen, nicht verändern, gibt man

`vi -R` *dat* oder `view` *dat*

ein. Zum Bearbeiten einer bereits vorhanden oder neu zu erstellenden Datei gibt man

`vi` *dat*

ein. Die Textdatei wird im Terminalfenster gezeigt, wobei nach dem Dateiende Zeilen erscheinen, die mit ∼ beginnen und nicht Teil der Datei sind, sondern den Platz zur untersten Zeile, der Statuszeile füllen. Nach dem Öffnen sieht man in der Statuszeile meistens den Namen der Datei und deren Anzahl an Zeilen (`L`) und Zeichen (`C`).

Im Gegensatz zu anderen Editoren gibt es beim *vi* zwei Betriebsarten (Modi), den Befehlsmodus (command mode) und den Schreibmodus (insert mode). Eigentlich ist noch ein erweiterter Befehlsmodus (command line mode) zu unterscheiden, aber davon sehen wir hier, der Einfachheit halber, ab. Im Befehlsmodus, mit dem man beginnt, kann man den Cursor im Text bewegen, Wörter löschen, in den Schreibmodus wechseln, speichern, und so weiter. Zu beachten ist Groß- oder Kleinschreibung. Im Schreibmodus kann man den Cursor bewegen (falls die Pfeiltasten der Tastatur funktionieren), Text schreiben und in den Befehlsmodus zurückkehren. Ist man nicht sicher, in welchem Modus man ist, gelangt man in jedem Fall durch Drücken der <Esc>-Taste in den Befehlsmodus.

Befehle im Befehlsmodus (weniger wichtige Befehle sind eingerückt)

← ↓ ↑ → (Pfeiltasten) oder `h j k l` bewegen den Cursor im Text

<Bild↓> oder <Strg>`F` bewegt den Cursor eine Seite (Fensterhöhe) nach unten

<Bild↑> oder <Strg>`B` bewegt den Cursor eine Seite (Fensterhöhe) nach oben

`w` springt mit dem Cursor zum Anfang des folgenden Worts

`b` springt mit dem Cursor zurück zum vorherigen Wortanfang

`:`*n* springt mit dem Cursor zum Anfang der *n*-ten Zeile

`G` springt mit dem Cursor zum Anfang der letzten Zeile

`/`*string*<Return> springt zur Zeichenkette *string*, erneute Suche mit `/`<Return>

man kann auch mit einem regulären Ausdruck suchen, `/[Jj]a` passt auf `Ja` und `ja`

`?`*string*<Return> sucht rückwärts nach *string*, erneute Suche mit `?`<Return>

`i` wechselt in den Schreibmodus, eingefügt wird vor der aktuellen Cursorposition

`a` wechselt in den Schreibmodus, eingefügt wird nach aktueller Cursorposition

`o` wechselt in den Schreibmodus, eingefügt wird in neuer Zeile unter dem Cursor

`s` wechselt in den Schreibmodus, ersetzt wird das Zeichen unter dem Cursor

`x` löscht das Zeichen unter dem Cursor und überträgt es in die Zwischenablage

`dd` löscht die aktuelle Zeile (mit Cursor) und überträgt sie in die Zwischenablage

`dG` löscht alle Zeilen von der aktuellen Zeile (einschließlich) bis zum Ende

`d1G` löscht alle Zeilen vom Anfang bis zur aktuellen Zeile (einschließlich)

`J` Zusammenfügen (englisch: join) der aktuellen und der folgenden Zeile

`u` Rücknahme eines Befehls, im *vi* nur für den letzten Befeh, im *vim* auch für mehrere

`U` Versetzt die aktuelle Zeile in den Zustand vor Positionierung des Cursors darin

`yy` kopiert die aktuelle Zeile in die Zwischenablage

n`yy` kopiert die aktuelle Zeile und *n*-1 folgende in die Zwischenablage

`yw` kopiert alle Zeichen vom Cursor bis zum Wortende in die Zwischenablage

n`yw` kopiert von Cursor bis Wortende und *n*-1 folgende Wörter in Zwischenablage

`p` fügt den Inhalt der Zwischenablage nach der Cursorposition ein

`:r` *dat2* <Return> fügt den Inhalt der Datei *dat2* nach der aktuellen Zeile ein

`:%s/`*war*`/`*love*`/g` ersetzt im ganzen Text (g = global) die Zeichenkette *war* durch *love*

`:set nu`<Return> bewirkt die Anzeige von Zeilennummern am Zeilenanfang

`:set nonu`<Return> bewirkt, dass keine Zeilennummern angezeigt werden

`:set tabstop=`*n*<Return> setzt die Breite der Tabulatortaste auf *n* Leerzeichen

`:set mouse=a`<Return> in *vim* (nicht *vi*) Nutzung der Maus in XTerm oder GUI

`ZZ` oder `:wq`<Return> oder `:wq!`<Return> speichert und beendet *vi*

`:w`<Return> oder `:w!`<Return> speichert alle Änderungen

(`:w!`<Return> speichert schreibgeschützte Datei, darf man den Schutz aufheben)

`:q`<Return> beendet *vi*, aber nur, wenn alle Änderungen gespeichert wurden

`:q!`<Return> beendet *vi*, auch wenn nicht alle Änderungen gespeichert wurden

<Esc> löscht einen angefangenen Befehl und beginnt den Befehlsmodus erneut

Befehle im Schreibmodus

← ↓ ↑ → (Pfeiltasten) bewegen den Cursor im Text (im *vi* manchmal nicht wirksam)

<Bild↓> <Bild↑> bewegt Cursor eine Seite (Fensterhöhe) nach unten oder oben

<Backshift> die Zeichenlöschtaste ist im *vi* meistens unwirksam, wirkt im *vim*

<Esc> beendet den Schreibmodus und kehrt zum Befehlsmodus zurück

4.2.3 Tastatureingabe von akzentuierten Sonderzeichen

Eine Compose-Taste ermöglicht es, Sonderzeichen in der Konsole erzeugen, die nicht auf der Tastatur abgebildet sind. Die Tastenkombination `Ctrl` `.` , also gleichzeitiges Drücken der `Ctrl`-Taste (Strg) und der Taste für den Punkt (.), wirkt als Compose-Taste. Alternativ kann man eine einzelne Taste als Compose-Taste konfigurieren (siehe Seite 43).

Drücken der Compose-Taste (1. Taste), danach Drücken einer Taste (oder Tastenkombination) für ein diakritisches Zeichen (2. Taste) und danach Drücken einer Buchstabentaste (3. Taste) erzeugt einen akzentuierten Buchstaben oder einen Umlaut. Die „2. Taste“ schließt Tastenkombinationen ein, die einen Akzent bilden. Zum Beispiel wird eine Tilde auf meiner Tastatur durch die Kombination der beiden Tasten `Alt Gr` und `* +` ∼ erzeugt. Das funktioniert unter anderem mit den in Tabelle 4.3 gezeigten Kombinationen. Manche akzentuierte Zeichen erscheinen erst, nachdem ein weitere Zeichen (zum Beispiel ein Leerzeichen) geschrieben wurde.

Akzent	1. Taste	2. Taste	3. Taste	Zeichen	Beispiel
Akut	Compose-Taste	´	e	é	thé ou café
Cedille	Compose-Taste	,	c	ç	façade
Gravis	Compose-Taste	`	e	è	mon père
Ringakzent	Compose-Taste	o	a	å	Håkon
Tilde	Compose-Taste	~	n	ñ	España
Zirkumflex	Compose-Taste	^	a	ñ	château
Umlaut	Compose-Taste	"	A	Ä	Ärger
Umlaut	Compose-Taste	"	o	ö	böse

Tabelle 4.3: Akzentuierte Sonderzeichen mit der Compose-Taste in der Konsole

4.3 Shellscripts mit der bash

Mit einer Shell (zum Beispiel der Bash) geschriebene Programme heißen Shellscripts. Die versteckte Datei `.bashrc` enthält ein Script, das beim Start der Bash ausgeführt wird. Mit diesem kann man eigene Einstellungen vornehmen oder Programme starten (siehe Abschnitt 2.1.3). Dies wirkt sich auch auf Shellscripts aus.

Es ist gut, Shellscripts in einem besonderen Verzeichnis, zum Beispiel `shell` zu speichern. Dafür erzeugt man im Terminal mit dem Befehl

```
mkdir /home/ahg/shell
```

ein Unterverzeichnis `shell` im Home-Verzeichnis `/home/ahg`.

Nach dem Speichern des Codes in einer Datei *datnam* muss man dem Betriebssystem mitteilen, wer das Script ausführen darf. Sollen alle Nutzer das Recht der Ausführung bekommen, gibt man den Befehl

```
chmod a+x datnam
```

im Terminalfenster ein.

Zur Ausführung eines Shellscripts kann der der Pfad zur Datei angegeben werden. Wenn das Shellscript im Verzeichnis `shell` ist und wir den Pfad zu diesem Verzeichnis in der Umgebungsvariablen `PATH` ergänzt haben, wie im Abschnitt 2.1.3 auf Seite 50 beschrieben, können wir Shellscripts nur mit dem Dateinamen aufrufen.

Oben (auf Seite 110) wurde erläutert, warum es sinnvoll ist, die Shebang-Zeile

```
#!/usr/bin/env bash
```

an den Anfang eines bash-Scripts zu setzen.

Unix-ähnliche Betriebssysteme, wie unser Linux, haben starke Hilfsprogramme zur Zeichenkettenverarbeitung. insbesondere `grep`, `sed` und `awk`, welche gut in Shellscripten verwendet werden können. Ein Beispiel für `sed` ist die Shellscripte zur Auswertung einer Webseite `curl` und `sed` auf Seite 153.

4.3.1 Grundlagen der bash-Programmierung

here documents

In einem Shellscript kann man Befehle geben, die den Inhalt einer Datei lesen und ihn dann irgendwie verarbeiten oder auf dem Bildschirm ausgeben. Manchmal ist es nützlich, diesen Inhalt innerhalb eines Shellscripts zu schreiben, statt in einer externen Datei. Das kann in einem *here document* (kurz heredoc) geschehen. Im ersten Beispiel werden mit dem `cat` Befehl zwei Zeilen auf die Standardausgabe (den Bildschirm) ausgegeben.

```
#!/usr/bin/env bash
var="Hallo"
cat <<'MEINE_MARKE'
Es folgen Sonderzeichen:
# \$ $var $(whoami)
MEINE_MARKE
```

nach dem Befehl `cat` und unmittelbar nach dem Zeichenpaar `<<` steht die die vorher willkürlich gewählte Zeichenkette (*limit string*) `MEINE_MARKE`, die kein Leerzeichen enthalten darf, und in der folgenden Zeile beginnt der Inhalt des (hier zweizeiligen) here documents. Es wird mit dem limit string beendet. Der limit string sollte so gewählt werden, dass er nicht im Inhalt des here documents enthalten ist. Bei der Ausführung des Shellscripts werden zwei Zeilen auf den Bildschirm geschrieben:

```
Es folgen Sonderzeichen:
# \$ $var $(whoami)
```

Wenn der limit string, wie in obigem Beispiel, in einfachen Anführungszeichen steht, verlieren die Sonderzeichen `\`, `$` und `‘` (backtick `` ` ``) ihre besondere Bedeutung im Shellscript und werden wie normale Zeichen ausgegeben. Dadurch wird $var als Zeichenkette $var ausgegeben und nicht durch den Wert der Variablen ersetzt. Das Doppelkreuz `#` ist aber in jedem here document (auch wenn der limit string nicht in einfachen Anführungszeichen steht) ein normales Zeichen, leitet also keinen Kommentar ein.

In folgendem zweiten Beispiel werden mit dem `cat` Befehl zwei Zeilen geschrieben, wobei die Ausgabe in die Datei `meintext.txt` umgeleitet wird. (Allgemein gilt: Nach dem limit string kann eine Ausgabeumleitung oder eine pipe folgen.) Die beiden Zeilen stehen in einem here document, dessen Ende wieder durch den limit string `MEINE_MARKE` markiert wird.

```
#!/usr/bin/env bash
var="Hallo"
cat <<MEINE_MARKE > meintext.txt
Es folgen Sonderzeichen:
# \$ $var $(whoami)
MEINE_MARKE
```

Da der limit string diesmal nicht in einfachen Anführungszeichen steht, behalten die Sonderzeichen \, $ und ‘ ihre besondere Bedeutung. Das Shellscript erzeugt die Datei `meintext.txt` (oder überschreibt eine bereits existierende Datei dieses Namens) . Die Datei erhält so den Inhalt

```
Es folgen Sonderzeichen:
# $ Hallo ahg
```

Dabei werden beide Zeilen mit einem Zeilenvorschub abgeschlossen. Will man den Inhalt des here documents an den Inhalt einer Datei anhängen (statt letztere zu überschreiben), wählt man `>>` statt `>` als Ausgabeumleitungsoperator.

Wenn man bei der Festlegung des limit strings einen Bindestrich - zwischen `<<` und den limit string setzt (ohne Leerzeichen), werden Tabulatorzeichen (horizontal tabulations), die im Inhalt des here documents am Zeilenanfang stehen, nicht an den Befehl weitergegeben. Damit wird es möglich, durch Einrückung des here documents das Shellscript besser lesbar zu machen. In folgendem Beispiel

```
#!/usr/bin/env bash
cat <<-MEINE_MARKE
        erstens kommt es anders und
        zweitens als man denkt
MEINE_MARKE
```

werden mit dem `cat` Befehl zwei Zeilen ohne Einrückung auf den Bildschirm geschrieben:

```
erstens kommt es anders und
zweitens als man denkt
```

Eine einfache Variante des here documents ist der *here string*. Damit können wir eine Zeichenkette statt einer Datei als Eingabe an einen Befehl senden. Das Shellscript

```
#!/usr/bin/env bash
wer="Welt"
cat <<< "Hallo ${wer}!"
```

ergibt auf dem Bildschirm die Ausgabe von

```
Hallo Welt!
```

Man beachte, dass der Operator `<<<` ist und ein limit string nicht gebraucht wird. Außerdem darf zwischen `<<<` und der folgenden Zeichenkette ein Leerzeichen stehen.

4.3.2 Kontrollstrukturen und Funktionen

Will man die Ausführung von Befehlen von einer Bedingung abhängig machen (die erfolgreiche Ausführung eines bestimmten Befehls *Bef* oder ob ein Ausdruck *A* als wahr erkannt wird), verwendet man eine if-Verzweigung oder eine case-Verzweigung.

In einer Schleife werden Befehle wiederholt abgearbeitet, bis eine Abbruchbedingung erfüllt ist. Dabei können sich in jedem Schleifendurchlauf Variablen verändern.

if-Verzweigung

Die Syntax der if-Verzweigung erlaubt verschiedene Schreibweisen, wir beschränken uns auf eine. Wir betrachten folgende Typen:

1. if *Bef1* ; then *Bef2* ; fi
2. if ! *Bef1* ; then *Bef2* ; fi
3. if [*A*] ; then *Bef* ; fi
4. if [*A*] ; then *Bef1* ; else *Bef2* ; fi
5. `if [` *A1* `] ; then` *Bef1* `; elif [` *A2* `] ; then` *Bef2* `; else` *Bef3* `; fi`

Zu beachten sind die Leerzeichen zwischen den Zeichenketten der Befehlszeile, auch vor und nach den eckigen Klammern (Zeichen `[` und `]`). Für den Vergleich von Zeichenketten wird der Ausdruck *A* in doppelte eckige Klammern eingefasst (siehe unten). Statt eines Semikolons kann ein Zeilenvorschub (newline) als Trennzeichen gesetzt werden. Einrückungen am Anfang einer Zeile beeinflussen das Programm nicht, können aber zur Lesbarkeit beitragen. Nach `then`, `elif` und `else` können statt eines Befehls mehrere folgen, die durch ein Trennzeichen (Semikolon oder Zeilenvorschub) getrennt sind.

In Typ 1 wird Befehl *Bef1* ausgeführt. War *Bef1* erfolgreich (exit status = 0), wird Befehl *Bef2* ausgeführt. Ansonsten (exit status $\neq$ 0) wird *Bef2* nicht ausgeführt. War ein Befehl nicht erfolgreich, ist sein exit status eine natürliche Zahl zwischen 1 und 255, die Aufschluss darüber geben kann, warum der Befehl nicht erfolgreich war (error code).

Eine verkürzte Schreibweise für Typ 1 ist *Bef1* `&&` *Bef2*

In Typ 2 wird Befehl *Bef1* ausgeführt. Befehl *Bef2* wird genau dann ausgeführt, wenn *Bef1* nicht erfolgreich war.

Eine verkürzte Schreibweise für Typ 2 ist *Bef1* `||` *Bef2*

In Typ 1 und Typ 2 kann anstelle von Befehl *Bef1* der Aufruf einer Funktion stehen, deren Exit-Staus dann gegprüft wird, siehe Seite 146.

In Typ 3 wird geprüft, ob der Ausdruck *A* wahr ist. Nur dann wird Befehl *Bef* ausgeführt.

Im Ausdruck *A* kann der Zahlenwert einer Variablen `x` mit dem einer Variablen `y` verglichen werden oder getestet werden, ob eine Variable `x` ungleich 0 ist.

- if [$x `-eq` $y] ; then *Bef* ; fi
- if [$x `-ne 0`] ; then *Bef* ; fi

Weitere Vergleichsoperatoren für Zahlen sind `-lt` (less than), `-gt` (greater than), `-le` (less or equal) und `-ge` (greater or equal).

Beim Vergleich von Zeichenketten werden andere Operatoren eingesetzt. Im Ausdruck *A* kann eine Variable auf Gleichheit (Operator `=`) oder Ungleichheit (Operator `!=`) mit einer Zeichenkette oder einem Zeichenkettenmuster getestet werden. Außerdem kann man testen, ob eine Variable leer (Operator `-z`) ist oder nicht leer (Operator `-n`) ist.

- if ["$x" != "$y"] ; then *Bef* ; fi
- if ["$x" = "Hallo"] ; then *Bef* ; fi
- if [-n "$x"] ; then *Bef* ; fi

Dass der Vergleichsoperator für „ist gleich“ als einfaches Gleichheitszeichen geschrieben wird, hat historische Gründe. In der Bash kann man hier auch ein doppeltes Gleichheitszeichen verwenden (==). Das würde aber bei Verwendung des Codes in der älteren Bourne-Shell zu einem Fehler führen.

In der Bash kann man eine Zeichenkette mit einem Regulären Ausdruck (Zeichenkettenmuster) vergleichen. Im folgenden Beispiel

```
if [[ "$x" =~ ^[0-9]+$ ]] ; then Bef ; fi
```

wird geprüft, ob die Zeichenkette `$x` eine positive ganze Zahl darstellt. `[0-9]` stellt eine beliebige Ziffer dar und das + bedeutet: mindestens einmal das vorherstehende Muster (hier: Ziffer). ^ markiert den Anfang der Zeichenkette und $ ihr Ende.

In einem Zeichenkettenmuster steht .* für eine beliebige Zeichenkette, die auch leer sein darf oder möglicherweise einen Punkt am Anfang hat oder ein Leerzeichen enthält.

Weiterhin kann im Ausdruck *A* geprüft werden, ob es ein Objekt *file* gibt, welches eine Datei (file) ist oder ein Objekt *dir* gibt, welches ein Verzeichnis (directory) ist, oder ob eine Datei *file* gibt, welche schreibbar (writable) ist.

- if [-f *file*] ; then *Bef* ; fi
- if [-d *dir*] ; then *Bef* ; fi
- if [-w *file*] ; then *Bef* ; fi

Entsprechend wird mit `-e` geprüft, ob ein Objekt existiert (unabhängig davon, ob es eine Datei oder ein Verzeichnis oder etwas anderes ist). Mit `-s` wird geprüft, ob ein Objekt existiert und es nicht leer ist. Mit `-x` wird geprüft, ob eine Datei ausführbar ist.

In Typ 4 wird Befehl *Bef 2* genau dann ausgeführt, wenn der Ausdruck *A* nicht wahr ist (und Befehl *Bef 1* nicht ausgeführt wurde).

In Typ 5 wird Befehl *Bef 1* ausgeführt, wenn der Ausdruck *A 1* wahr ist. Befehl *Bef 2* wird ausgeführt, wenn *A 1* nicht wahr ist und der Ausdruck *A 2* wahr ist. Befehl *Bef 3* wird ausgeführt, wenn weder *A 1* noch *A 2* wahr sind.

Die logischen Operatoren `-a` (UND) und `-o` (ODER) verknüpfen zwei Vergleiche. Der Operator `!` (NICHT) negiert den folgenden Wahrheitswert im Ausdruck *A* (wahr → unwahr, unwahr → wahr). Die logischen Operatoren werden erst nach den Vergleichsoperatoren ausgewertet.

Im folgenden Beispiel prüft ein Shellscript, ob die Variable `x` einen Pfad zu einer Datei enthält und außerdem, ob der Dateiname auf `.mp3` oder `.wav` endet.

```
#!/usr/bin/env bash
x="$HOME/audio/TinoRossi-SantaLucia.mp3"
e=‘echo "$x" | rev  | cut -c 1-4 | rev‘
if [ -f "$x" ] && [ "$e" = ".mp3" -o "$e" = ".wav" ]
then echo "$x is an audio file" ; else echo "no" ; fi
```

Wird der Befehl `rev` ohne Dateinamen verwendet, liest er eine Zeichenkette von der Standardeingabe und gibt sie in umgekehrter Zeichenreihenfolge auf der Standardausgabe aus. Wird `cut -c 1-4` ohne Dateinamen verwendet, liest er eine Zeichenkette von der Standardeingabe und gibt die ersten vier Zeichen auf der Standardausgabe aus.

case-Verzweigung

Oft soll der Wert einer Variablen darüber entscheiden, welcher von mehreren Befehlen ausgeführt wird. Dies kann mit verschachtelten if-Verzweigungen programmiert werden, aber eine kürzere und besser lesbare Lösung ist eine case-Verzweigung. Sie hat die Form

```
case "$x" in
Str 1) Bef 1 ;;
Str 2) Bef 2 ;;
*) Bef 3 ;;
esac
```

Wenn die Variable `$x` die Zeichenkette *Str 1* als Wert hat, wird Befehl *Bef 1* ausgeführt. Wenn die Variable `$x` *Str 2* als Wert hat, wird *Bef 2* ausgeführt. Wenn die Variable `$x` keine der vorgenannten Zeichenketten als Wert hat, wird *Bef 3* ausgeführt. Die Zeilen mit möglichen Zeichenketten (hier: zwei) können beliebig erweitert werden. Statt Zeichenketten können auch Zeichenkettenmuster eingesetzt werden.

Die case-Verzweigung funktioniert auch in der Bourne-Shell, einer früheren Shell als die hier verwendeten Bash. In der Bourne-Shell würde ein Fehler auftreten, wenn die Variable `$x` leer ist und nicht in Anführungszeichen eingefasst wäre. Man vermeidet diesen Fehler, indem die Variable in doppelte Anführungszeichen gesetzt wird: `"$x"`. Wenn man die Variante mit den Anführungszeichen benutzt, ist der gleiche Code daher mit verschiedenen Shells verwendbar.

for-Schleife

Die for-Schleife führt für alle Elemente einer Liste einen Block aus, also eine Folge von Befehlen. Sie hat die kurze Form

`for x in` *liste* `do` *Bef 1* `;` *Bef 2* `; done`

Meistens verwendet man eine längere, aber besser lesbare Schreibweise:

```
for x in liste
do
  Bef 1
```

Bef2
`done`

Der Befehlsblock liegt zwischen den Schlüsselwörtern `do` und `done`. Die Anzahl der Befehle im Block ist beliebig. Eine Einrückung am Zeilenanfang ist für das Programm unbedeutend, kann aber die Lesbarkeit verbessern.

Die Elemente der Liste (Zeichenketten) werden durch Trennzeichen getrennt. Welche das sind, bestimmt die Umgebungsvariable `IFS` (Internal Field Separator). Voreingestellt ist: beliebig viele Leerzeichen, Tabs oder Zeilenvorschübe.

Die Liste kann durch explizite Aufzählung gebildet werden. Beispiel:

```
for x in datei1 datei2 datei3
do
 cp $dat $dat.backup
done
```

Jede der drei Dateien in der Liste wird auf eine Datei mit neuem Namen kopiert. In `$dat.backup` endet der Variablenname vor dem Punkt, weil ein Punkt nicht Teil eines Variablennamens sein kann. Die Verkettung des Variablennamens mit der folgenden Zeichenkette (`.backup`) erfolgt durch Hintereinanderschreiben ohne Lücke.

Die Liste kann durch Dateinamen-Expansion mithilfe von Wildcards (∗, ? oder eine in eckigen Klammern eingeschlossene Zeichenmenge [. . .]) gebildet werden. Beispiel:

```
for x in *  .*
do
 if [ -f $x ] ; then echo "$x" ; fi
done
```

Dabei wird `*` durch alle Datei- und Verzeichnisnamen ersetzt, die nicht mit einem Punkt beginnen und `.*` durch alle, die mit einem Punkt beginnen. Wenn `$x` eine Datei ist, wird ihr Name auf dem Bildschirm ausgegeben. Es werden also alle Dateinamen des aktuellen Verzeichnisses ausgegeben, erst die nicht versteckten, dann die nicht versteckten.

Die Liste kann auch durch Kommando-Substitution erzeugt werden. Im folgenden Beispiel wird die Ausgabe des Befehls `cat datei` zur Liste.

```
for x in ‘cat datei‘
do
 echo $x
done
```

Der Befehl `cat datei` gibt die Zeilen einer Datei `datei` aus. Die einzelnen Worte werden im Block auf den Bildschirm geschrieben. (Man könnte Spannenderes damit machen.)

while-Schleife und until-Schleife

Die while-Schleife führt einen Befehlsblock immer wieder aus, solange ein bestimmter Befehl erfolgreich ausgeführt wird oder eine Aussage wahr ist. Sie hat also die Formen

1. while *Bef1* ; do *Bef2* ; *Bef3* ; done

2. while [*A*] ; do *Bef2* ; *Bef3* ; done

Der Befehlsblock besteht hier aus *Bef2* und *Bef3*) und der Befehl dessen erfolgreiche Ausführung geprüft wird, ist *Bef1*. Zu beachten sind die Leerzeichen zwischen den Zeichenketten der Befehlszeile, auch vor und nach den eckigen Klammern (Zeichen `[` und `]`). Statt eines Semikolons wird meistens ein Zeilenvorschub (newline) als Trennzeichen gesetzt. Einrückungen am Anfang einer Zeile beeinflussen das Programm nicht, können aber zur Lesbarkeit beitragen. Der Befehlsblock, der oben aus zwei Befehlen (*Bef2* und *Bef3*) besteht, kann beliebige viele Befehle enthalten, die durch ein Trennzeichen (Semikolon oder Zeilenvorschub) getrennt sind.

Manchmal will man einen einen Befehlsblock immer wieder ausführen, solange ein bestimmter Befehl *nicht* erfolgreich war. Dies kann mit einer while-Schleife und dem logischen Operator `!` (NICHT) vor *Bef1* geschehen. Alternativ kann man das Schlüsselwort *until* nehmen. Die beiden foolgenden Formen verhalten sich also gleich.

3. while ! *Bef1* ; do *Bef2* ; *Bef3* ; done

4. until *Bef1* ; do *Bef2* ; *Bef3* ; done

Zu beachten ist, dass in den Formen 1, 3 und 4 immer versucht wird, den Befehl *Bef1* auszuführen; der folgende Befehlsblock wird jedoch nur ausgeführt, wenn *Bef1* erfolgreich war (Form 1), beziehungsweise nicht erfolgreich war (Formen 3 und 4).

Mit einer while-Schleife kann ein Zahlenbereich durchlaufen werden. Im folgenden Beispiel wird die Summe der Zahlen von 1 bis 10 gebildet.

```
typeset -i zahl ende summe
zahl=1 ; ende=10 ; summe=0
while [ $zahl -le $ende ]
do
 summe=$summe+$zahl
 zahl=$zahl+1
done
echo $summe
```

Mit dem Befehl `typeset -i zahl ende summe` werden die Variablen `zahl`, `ende` und `summe` als ganze Zahlen (integers) deklariert. Nun kann mit ihnen gerechnet werden. Der Vergleichsoperator `-le` bedeutet „kleiner oder gleich“.

Im folgenden Beispiel wird eine Eingabe vom Nutzer abgefragt (sein Name) und daraufhin eine Begrüßung ausgegeben. Das Ganze wiederholt sich in einer until-Schleife, bis `quit` oder `q` eingegeben wird. (Es gibt lustigere Programme.)

```
echo "Abbruch durch Eingabe von quit oder q"
echo -e "Name? \c" ; read inp
until [ "$inp" = "quit" -o "$inp" = "q" ]
do
 echo "Hallo $inp"
```

```
  echo -e "Name? \c" ; read inp
done
```

Die Option `-e` des echo-Befehls bewirkt die Auswertung von Escape-Squenzen (Zeichenketten, die mit einem Backslash \ beginnen). Damit unterdrückt `\c` den Zeilenvorschub, sodass die Antwort des Nutzers in der gleichen Zeile erfolgt. `"$inp"` wurde in Anführungszeichen gesetzt, damit auch Nutzereingaben, die Leerzeichen enthalten, richtig verarbeitet werden. Zum Vergleichsoperator `=` siehe Seite 140. Zwischen zwei Zeichenkettenvergleichen steht `\-o` als logischer Operator für eine ODER-Verknüpfung.

Endlosschleifen und break

Wenn nach `while` ein Befehl steht, der immer erfolgreich ist (exit status = 0), entsteht eine Endlosschleife. Der Befehl `true` tut nichts, hat aber immer Erfolg (wie mein Onkel Felix). Eine andere Schreibweise für `true` ist `:` (in der Shell). Beispiel:

```
while : ; do echo "Wälz den Stein nach oben, Sisyphos!" ; done
```

Nur der Tod des Programms kann es nun noch stoppen, zum Beispiel durch `Ctrl-C` (siehe Seite 12). Man kann jedoch in allen Schleifen, auch in Endlosschleifen, einen Abbruchbefehl einbauen, der unter einer festgesetzten Bedingung ausgeführt wird: `break`. Mit `break` verlässt man die Schleife und die Programmausführung setzt mit den Befehlen nach der Schleife fort.

Im folgenden Beispiel wird das oben (auf Seite 143) beschriebene Programm mit Nutzereingabe und Begrüßung mit einer break-Anweisung neu geschrieben.

```
echo "Abbruch durch Eingabe von quit oder q"
while true
do
  echo -e "Name? \c" ; read inp
  [ "$inp" = "quit" -o "$inp" = "q" ] && break
  echo "Hallo $inp"
done
```

Der Vorteil dieser Programmversion ist, dass die Programmzeile mit der Nutzereingabe nur einmal vorkommt. Die if-Verzweigung mittels `&&` wurde auf Seite 139 behandelt.

Funktionen

Funktionen bilden einen benannten Befehlsblock. Man kann damit bestimmte Aufgaben übersichtlich kapseln und sie unter ihrem Namen wiederholt ausführen lassen. Zunächst wird eine Funktion definiert.

fn`() { `*Bef1*` ; `*Bef2*` ; }`

Wenn der Code auch in der älteren Bourne-Shell fehlerfrei laufen soll, darf der Funktionsname (*fn*) nicht gleichzeitig der Name einer Variablen sein. Die runden Klammern sind Teil der Funktionsdefinition. Sie können keine Argumente enthalten. Nach `{` muss zunächst ein Leerzeichen oder ein Zeilenvorschub stehen und vor `}` muss ein Semikolon

oder ein Zeilenvorschub stehen. Im Befehlsblock können beliebig viele Befehle sein, die durch Semikolon oder Zeilenvorschub getrennt werden.

Wurden einer Funktion keine Parameter übergeben, ruft man sie mit ihrem Namen auf und der definierte Befehlsblock wird abgearbeitet. Das folgende kleine Programm schreibt mithilfe einer Funktion `hw1` dreimal eine Zeichenkette auf den Bildschirm.

```
hw1() { echo "$x Welt!" ; }
x="Hallo"
hw1 ; hw1 ; hw1
```

Funktionen werden in der aktuellen Shell ausgeführt (keine Subshell). Daher sind alle Variablen des Programms normalerweise innerhalb und außerhalb von Funktionen les- und schreibbar (globale Variablen). Das betrifft im Beispiel die Variable `x`. Ausnahmen bilden die beim Aufruf übergebenen Positionsparameter und die Variable FUNCNAME (siehe unten).

In der Bash (nicht aber in der älteren Bourne-Shell) kann man zusätzlich in einer Funktionsdefinition lokale Variablen erzeugen, die außerhalb der Funktion keine Bedeutung haben. Im folgenden, wenig tiefsinnigen Beispiel

```
hw2() { local x="Welt!" ; echo "Hallo $x" ; }
hw2 ; hw2 ; hw2
if [ -z $x ] ; then echo "Es gibt keine globale Variable x." ; fi
```

wird wieder dreimal `Hallo Welt!` auf den Bildschirm geschrieben. Auf die Variable `x` kann nicht außerhalb der Funktion zugegriffen werden.

Die Übergabe von Argumenten geschieht beim Funktionsaufruf durch eine Liste mit dem Leerzeichen als Trennzeichen. Im Befehlsblock der Funktion wird mit den Positionsparametern $1, $2 ... auf die entsprechenden Argumente zugegriffen. Die Positionsparameter haben also in der Funktion eine andere Bedeutung als außerhalb (wo sie sich auf Kommandozeilen-Parameter beziehen).

```
hw3() { echo "Hallo ${1}!" ; }
hw3 Anton ; hw3 Berta ; hw3 Caesar
```

Schließt sich an einen Positionsparameter ein Zeichen an, das Teil eines Variablennamens sein könnte, fasst man seinen Namen in geschweifte Klammern ein, so wie auch bei anderen Variablen üblich. (In obigem Beispiel war das allerdings unnötig, da `!` nicht Teil eines Variablennamens sein kann.)

In der Bash (nicht aber in der älteren Bourne-Shell) kann man innerhalb einer Funktion mit der Variablen FUNCNAME auf den Namen der Funktion zugreifen.

```
hw4() { echo "Funktion $FUNCNAME" ; } ; hw4
```

Die Ausgabe des Namens einer laufenden Funktion kann bei der Fehlersuche helfen.

In den meisten Programmiersprachen kann einer Variablen der Rückgabewert eines Funktionsaufrufs zugewiesen werden. In einem Shellscript geschieht kann dies über den Umweg der Kommandosubstitution erfolgen.

```
hw5() { local x="Hallo $1" ; echo $x ; }
echo -e "`hw5 Anton` \c" ; y=`hw5 Berta` ; echo $y
```

ergibt die Ausgabe `Hallo Anton Hallo Berta`. Die Option `-e` des echo-Befehls bewirkt die Auswertung von Escape-Squenzen (Zeichenketten, die mit einem Backslash \ beginnen). Damit unterdrückt `\c` den Zeilenvorschub, sodass nach `Hallo Anton` kein Zeilenvorschub erfolgt.

Eine einfache Möglichkeit zur Rückgabe eines Werts aus einer Funktion ist die Änderung einer Variablen

```
hw6() { x="Berta" ; }
x="Anton" ; echo "Hallo ${x}!"
hw6 ; echo "Hallo ${x}!"
```

Das geht, weil die Variablen des Programms global sind, also auch in Funktionen gelten.

Jede Funktion gibt nach der Abarbeitung ihres Befehlsblocks einen Wert direkt an das Programm zurück, nämlich den Exit-Status. Der Exit-Status einer Funktion ist, wie bei einem gewöhnlichen Befehl, eine Zahl zwischen 0 und 255. 0 signalisiert die erfolgreiche Ausführung, während alle anderen Werte einen Fehler bedeuten. Jede Funktion kann daher, wie ein gewöhnlicher Befehl, in einer if-Verzweigung geprüft werden.

Wie wird der Exit-Status erzeugt? Im Befehlsblock einer Funktion beendet der Befehl `return` n die Funktion, wobei n der Exit-Status ist. Ohne return-Befehl, oder mit `return` ohne n, ist der Exit-Status der Funktion gleich dem Exit-Status des zuletzt ausgeführten Befehls. Folgendes Programm enthält zwei return-Befehle mit verschiedenen Exit-Codes.

```
hw7() {
if [ -f datei.txt ] ; then return 0 ; fi
return 1 ; }
if hw7
  then echo "datei.txt vorhanden"
  else echo "datei.txt nicht vorhanden"
fi
```

Die if-Verzweigung zur Prüfung, ob es eine bestimmte Datei gibt, und die if-Verzweigung mit `else` wurden bereits erläutert (siehe Seite 140 beziehungsweise Seite 140).

Während `return` nur den Befehlsblock der Funktion beendet, würde `exit` das Shellscript beenden, innerhalb wie außerhalb einer Funktion.

4.3.3 Nützliche Shellscripts

Für Information über oder Änderungen an der Funktion des Betriebssystems oder andere Aufgaben kann man besondere Shellscripts einsetzen. Es folgen einige Beispiele.

System-Information

Information zu Rechner und Betriebssystem wird von folgendem Script ausgegeben.

```
#!/usr/bin/env bash
# sysinfo.sh - bash script for system information (AHG, 2021)
sernr=$(cat /proc/cpuinfo | grep  Serial | cut -d' ' -f2) # serial nr.
```

```
sw=$(cat /sys/class/graphics/fb0/virtual_size | cut -d, -f1) # width
sh=$(cat /sys/class/graphics/fb0/virtual_size | cut -d, -f2) # height
ftf=$(grep "FONTFACE" /etc/default/console-setup) # font in console
fts=$(grep "FONTSIZE" /etc/default/console-setup) # font in console
mcd=$( df -h | grep "/dev/mmcblk0p2" | awk '{print $2}' ) # memory of /
mfr=$( df -h | grep "/dev/mmcblk0p2" | awk '{print $4}' ) # free memory
vma=$(vcgencmd get_mem arm) # cpu memory
vmg=$(vcgencmd get_mem gpu) # gpu memory
echo -e "\nsystem information\n------------------"
date "+%A, %B %-d, %Y, %H : %M"
echo -e "computer: \c" ; cat /proc/device-tree/model
echo -e "\nserial number: ${sernr}"
echo "operating system: $(hostnamectl \
| awk -F': ' 'NR==5 {print $2}')"
echo "hostname: $( hostnamectl | grep "hostname" \
| awk '{print $3}' ), actual user: $(whoami)"
echo "home directory: $HOME"
nmq=`nmcli connection show --active`
if echo "$nmq" | grep -q "eth0"; then
echo "ethernet eth0 is active, IP address: $(hostname -I \
| awk '{print $1}')"
else
echo "ethernet eth0 is not active"
fi
echo "eth0 MAC address: $(cat /sys/class/net/eth0/address)"
if echo "$nmq" | grep -q "wlan0"; then
echo "wifi wlan0 is active, IP address: $(hostname -I \
| awk '{print $1}')"
essid=`iw dev wlan0 link | sed -n '2 p' | awk '{print $2}'`
wlanf=`iw dev wlan0 link | sed -n '3 p' | awk '{print $2}'`
wlpsk=`nmcli device wifi show-password | grep Password | \
awk '{print $2}'`
echo "wifi ESSID: $essid, freq.: $wlanf MHz, psk: $wlpsk"
else
echo "wifi wlan0 is not active"
fi
echo "bluetooth MAC address: $( hcitool dev |
grep "hci0" | awk '{print $2}' )"
echo "language (environment variable LANG): $LANG"
echo "screen resolution: width ${sw}, height ${sh}"
echo "terminal font: ${ftf}, ${fts}"
echo "SD memory of / (Byte): total $mcd, free $mfr"
echo "volatile memory (Byte): $vma, $vmg"
echo "GPU: $(vcgencmd measure_temp), $gpv$(vcgencmd measure_volts)"
```

```
if [ -d /media/usb_1 -a -n "$(ls -A /media/usb_1)" ] ; then
echo "USB Datenträger an mount point /media/usb_1" ; fi
echo "framebuffer parameters:"
typeset i fb_width fb_height fb_bitspp
f_fbres="/sys/class/graphics/fb0/virtual_size"
f_fbbpp="/sys/class/graphics/fb0/bits_per_pixel"
fb_width=‘awk -F’,’ ’{print $1}’ $f_fbres‘
fb_height=‘awk -F’,’ ’{print $2}’ $f_fbres‘
fb_bitspp=‘cat $f_fbbpp‘
echo "fb_width = $fb_width, fb_height = $fb_height, \
fb_bitspp = $fb_bitspp"
echo
```

Die Ausgabe des Scripts sieht ungefähr so aus:

```
system information
------------------
Sunday, December 10, 2023, 20 : 27
computer: Raspberry Pi 4 Model B Rev 1.1
serial number: 10000000b3455f4a
operating system: Raspbian GNU/Linux 12 (bookworm)
hostname: ahg-desk, actual user: ahg
home directory: /home/ahg
ethernet eth0 is not active
eth0 MAC address: dc:a6:32:34:90:a9
wifi wlan0 is active, IP address: 192.168.178.137
wifi ESSID: greifgasse3, freq.: 5500 MHz, psk: passwort
bluetooth MAC address: DC:A6:32:34:90:AB
language (environment variable LANG): en_GB.UTF-8
screen resolution: width 1920, height 1080
terminal font: FONTFACE="Terminus", FONTSIZE="16x32"
SD memory of / (Byte): total 15G, free 9.7G
volatile memory (Byte): arm=948M, gpu=76M
GPU: temp=59.9’C, volt=0.8500V
framebuffer parameters:
fb_width = 1920, fb_height = 1080, fb_bitspp = 16
```

Im Befehl `echo -e "\nsystem information\n---------"` bewirkt die Option `-e` zusammen mit den beiden `\n` zwei Zeilenvorschübe, jeweils an der Stelle von `\n`.

Der date-Befehl gibt Datum und Zeit aus. Die zugehörige Format-Zeichenkette beginnt mit `+` und enthält `%A` (Wochentag), `%-d` (Tag des Monats ohne führende 0), `%B` (ganzer Monatsname), `%Y` (vierstellige Jahreszahl), `%H` (Stunde zwischen 0 und 23) und `%M` (Minute).

Im Befehl `echo -e "computer: \c" ; cat /proc/device-tree/model` verhindert die Option `-e`, zusammen mit `\c`, einen Zeilenvorschub. Danach wird durch den cat-Befehl der Inhalt einer Textdatei ausgegeben, der das Rechnermodell beschreibt, gefolgt von einem Nullzeichen, aber keinem Zeilenvorschub.

Im Befehl `echo -e "\nBetriebssystem ...` bewirkt `\n` einen Zeilenvorschub.

Durch Kommando-Substitution mittels `$(...)` wird das Ergebnis einer Zeichenkettenanalyse ausgegeben. Im Fall von $(hostnamectl | awk -F': ' 'NR==5 print $2') besteht die Analyse aus der Übertragung (mittels Pipe `|`) der Ausgabe des Befehls `hostnamectl` auf `awk` und die Verarbeitung und Ausgabe durch `awk`. Der hostnamectl-Befehl gibt in mehreren Zeilen Information des Betriebssystems aus. Der awk-Befehl extrahiert mit `NR==5` die fünfte Zeile der Ausgabe. Als Trennzeichen der Felder einer Zeile wird mit `-F': '` eine zweielementige Zeichenkette aus Doppelpunkt und Leerzeichen festgelegt. Damit extrahiert `$2` das zweite Feld der Zeile und gibt es mittels `print` aus.

Information zum Framebuffer

Die Größe des Framebuffers, Breite (width) und Höhe (height) in pixels (Bildpunkten) und die bits per pixel (Bits pro Bildpunktblock) erfährt man mit folgendem Script.

```
#!/usr/bin/env bash
# framebuffer parameters
typeset i fb_width fb_height fb_bitspp
f_fbres="/sys/class/graphics/fb0/virtual_size"
f_fbbpp="/sys/class/graphics/fb0/bits_per_pixel"
fb_width=‘awk -F’,’ ’{print $1}’ $f_fbres‘ ; echo $fb_width
fb_height=‘awk -F’,’ ’{print $2}’ $f_fbres‘ ; echo $fb_height
fb_bitspp=‘cat $f_fbbpp‘ ; echo $fb_bitspp
```

WLAN

Die empfangbaren Funknetzwerke und ihre Signalstärken werden mit dem Script

```
#!/usr/bin/env bash
# wlanscan (AHG, 2016)
sudo iw dev wlan0 scan > scan.tmp
e=‘grep ’SSID:’ scan.tmp | awk ’{print $2}’‘
z=‘echo $e | wc -w‘
f=‘grep ’freq:’ scan.tmp | awk ’{print $2}’‘
s=‘grep ’signal:’ scan.tmp | awk ’{print $2}’‘
i=1 ; echo
echo "Über wlan0 sind "${z}" Funknetzwerke erreichbar:"
while [ $i -le $z ]
do
    t1=${i}") ESSID: "‘echo $e | cut -d " " -f $i‘
    t2=", Frequenz: "‘echo $f | cut -d " " -f $i‘
```

```
    t3=", Signalstärke: "‘echo $s | cut -d " " -f $i \
| cut -d "." -f 1‘ ; echo ${t1}${t2}" MHz"${t3}" dBm"
    i=‘expr $i + 1‘
done
echo ; rm scan.tmp
```

angezeigt. Voraussetzung ist, dass der WLAN-Empfänger nach neuen Funknetzwerken suchen (scannen) kann, wenn er bereits verbunden ist, was nicht alle Empfänger können.

Die Signalstärke einer bestehenden WLAN-Verbindung wird mit dem Script

```
#!/usr/bin/env bash
echo "Shell-Skript wlansignal (AHG, 2016), Abbruch mit Ctrl-C"
echo -e "ESSID: ‘iw dev wlan0 link |
sed -n ’2 p’ | awk ’{print $2}’‘, \c"
echo -e "Frequenz: ‘iw dev wlan0 link |
sed -n ’3 p’ | awk ’{print $2}’‘ MHz, \c"
echo "Signalstärke (dBm):"
while true
do
    echo -e "\r‘iw dev wlan0 link | sed -n ’6 p’ | awk ’{print $2}’‘ \c"
done
```

kontinuierlich im Terminal ausgegeben und man kann die Antenne für optimalen Empfang ausrichten. Das Script wird mit Ctrl-C beendet (siehe Seite 12).

Menü

Verschiedene wiederkehrende Aufgaben möchte man ohne viel Schreibarbeit auslösen können. Dafür bietet sich ein einfaches Auswahlmenü an:

```
#!/usr/bin/env bash
# einfaches Menü (erweiterbar)
PS3="Auswahl (Zahl): " # Auswahl-Prompt
optarr=( # Array-Beginn
"uno"    # 1
"dos"    # 2
"tres"   # 3
)        # Array-Ende
while true
do
    echo ; echo "Menü - Überschrift"
    select opt in "${optarr[@]}" "Ende";
    do
        case "$REPLY" in
        1 ) echo "eins: $opt gewählt"; break ;;
```

```
        2 ) echo "zwei: $opt gewählt"; break ;;
        3 ) echo "drei: $opt gewählt"; break ;;
        $(( ${#optarr[@]}+1 )) ) echo ; exit;;
        *) echo "Unzulässig! (4 = Ende)"; continue;;
        esac
    done
done
```

MP3-Player mit zufälliger Dateiauswahl

Folgendes Script (`mp3loop.sh`) spielt endlos MP3-Musikdateien des vorgegeben Verzeichnisses `$HOME/audio` mit dem Mediaplayer `mplayer` (siehe Seite 186) ab.

```
#!/usr/bin/env bash
# mp3loop.sh (MP3 mit mplayer)
vz="$HOME/audio" # Verzeichnis
while true  # "Ctrl-C zum Abbruch"
do
    if [ -d $vz ] ; then
        find $vz -name '*.mp3' | shuf -n 1 | xargs -d '\n' m\
player # -ao alsa
    else
        echo "Das Verzeichnis $vz fehlt!"
        exit 1
    fi
done
```

In der Umgebumgsvariablen `$HOME` ist das Heimatverzeichnis des Nutzers gespeichert. Der Befehl `shuf` mischt die übergebenen Dateinamen und seine Option `-n 1` beschränkt die Ausgabe auf eine Zeile, also den ersten Dateinamen. Die Option `-d '\n'` des Befehls `xargs` bestimmt den Zeilenumbruch (newline) als Trennzeichen der Argumente, statt des Leerzeichens. Damit können auch Dateinamen weitergegeben werden, die ein Leerzeichen im Namen haben. Einen Zeilenumbruch dürfen die Dateinamen aber nicht enthalten. In obigem Programm wurde `mplayer` mit dem Audioausgabetreiber `alsa` verwendet. Wenn man PulseAudio benutzt, gibt man als Audioausgabetreiber `pulse` an.

Compilierung eines LaTeX Dokuments

Dateien mit LaTeX Quellcode müssen mit dem Programm `pdflatex` compiliert werden, um eine PDF-Datei zu erzeugen, siehe Seite 239.

Das folgende Script (`lc`) verkürzt den notwendigen Befehl zur Compilierung, unterdrückt lange Bildschirmausgaben während des Compilierens, löscht Hilfsdateien automatisch und zeigt die fertige PDF-Datei mithilfe des Programms `fbgs` (siehe Seite 179).

```
#!/usr/bin/env bash
```

```
# lc (latexcompile)
verz="/home/ahg/latex/"

if [ $# -eq 0 ]; then
    datei="/home/ahg/latex/bsp.tex" # Beispiel-Datei
elif [ $# -eq 1 ]; then
    datei=$1 # verwende die Datei im Argument
else exit 3 ; fi # zu viele ( > 1 ) Argumente

if [ ! -f $datei -a ! -f verz$datei ] ; then
    echo "FEHLER: Die angegebene Datei gibt es nicht !"
    exit 2
fi

basis="${datei%.*}" # der Dateiname ohne Endung
echo "Compiliere Datei $datei und zeige PDF (q -> Ende)."
pdflatex -output-directory $verz $datei > /dev/null 2>&1
if [ $? -ne 0 ]; then
     echo "FEHLER beim Compilieren, siehe ${basis}.log !"
     exit 1
fi

if [ -f "${basis}.log" ] ; then rm "${basis}.log" ; fi
if [ -f "${basis}.aux" ] ; then rm "${basis}.aux" ; fi
if [ -f "${basis}.out" ] ; then rm "${basis}.out" ; fi

fbgs -xxl ${basis}.pdf > /dev/null 2>&1
exit 0
```

Zum Beispiel wird mit dem Befehl

```
lc datei.tex
```

die Datei `datei.tex` compiliert. Ohne Angabe des Pfades zur Datei wird sie im aktuellen Arbeitsverzeichnis und im Verzeichnis `/home/ahg/latex/` gesucht.

Eigenen Standort und öffentliche IP Adresse bestimmen

Jeder Rechner, der Daten von einem Server aus dem Internet empfängt, hat eine sogenannte IP-Adresse, welche angibt, wohin der Server die Daten senden soll. Genauer gesagt, gibt es eine private IP-Adresse und eine öffentliche IP-Adresse.

Die private IP-Adresse identifiziert den Rechner im lokalen Netzwerk identifiziert. Sie wird dem Rechner vom Netzwerkrouter zugewiesen und sollte außerhalb des lokalen Netzwerks nicht sichtbar sein.

Der Netzwerkrouter des lokalen Netzwerks wird im Internet durch eine öffentliche IP-Adresse identifiziert, welche vom Internetdienstanbieter (internet service provider) zugewiesen wird.

Wenn ein Prozess (laufendes Programm) des Rechners eine Anfrage (request) an einen Server im Internet sendet, nennt man ihn einen *client*. Der Server kennt die öffentliche IP-Adresse des client und kann daraus den Standort des lokalen Netzwerks, und damit der Standort des Rechners ungefähr (aber nicht genau) bestimmen.

Das folgende Programm nutzt einen Server, der die öffentliche IP-Adresse und den ungefähren Standort an den client zurückmeldet.

```
#!/usr/bin/env bash
# public IP address and geographic location
#
# get the public IP address (limits apply)
p=$(curl -s https://ipinfo.io/ip)
#
# check and abort, if address not retrieved
if [ -z "$p" ] ; then
echo "ERROR: could not get public ip adress"
exit 1 ; fi
#
# get the geo location data (limits apply)
i=$(curl -s https://ipinfo.io/$p)
#
# extract city, latitude and longitude
c=$(echo $i | sed -nE 's/.*city": "([^"]*)".*/\1/p')
a=$(echo $i | sed -nE 's/.*loc": "([^,]*),.*/\1/p')
o=$(echo $i | sed -nE 's/.*loc": "[^,]*,([^"]*)".*/\1/p')
#
# print results
echo "public ip address: $p"
echo "             city: $c"
echo "         latitude: $a"
echo "        longitude: $o"
```

Webseite auswerten mit curl und sed

Wer Auto fährt, möchte gern wissen, bei welcher Tankstelle in der Umgebung der Treibstoff günstig ist. Dafür gibt es Webseiten, die den Preis bei verschiedenen Tankstellen vergleichen. Das folgende `bash`-Script schickt einen Ort als Postleitzahl an den Server von `https://www.clever-tanken.de/` und lädt die Webseite mit den gewünschten Informationen herunter (download).

Für den download wird das Programm `curl` verwendet. Wir verzichten auf eine Anzeige der Webseite mit einem Browser und ziehen die wichtigen Daten mit dem Programm `sed` heraus, um sie danach in Textform auf dem Bildschirm auszugeben.

```
#!/usr/bin/env bash
# local german gasoline price / Benzinpreis-Info
#
# Die folgenden 3 Zeilen müssen angepasst werden:
plz="07743" # Postleitzahl für Tankstellensuche
sorte="5" # Treibstoff: 5 = Super E10
radius="10" # Radius in km um Ort der Postleitzahl
#
if [ $sorte -eq 2 ] ; then t="LKW-Diesel" ; fi
if [ $sorte -eq 3 ] ; then t="Diesel" ; fi
if [ $sorte -eq 5 ] ; then t="Super E10" ; fi
if [ $sorte -eq 6 ] ; then t="SuperPlus" ; fi
if [ $sorte -eq 7 ] ; then t="Super E5" ; fi
#
# URL of server website with form for GET request
b="https://www.clever-tanken.de/tankstelle_liste?"
u="${b}ort=${plz}&spritsorte=${sorte}&r=${radius}"
#
# curl: transfers data from a server to the client
# curl uses HTTP(S) request method GET by default
# curl option -s # (silent) suppress progress bar
# curl option -o "FIL" # output to FIL not stdout
curl -s -o "tmp" $u # goto website, save response
#
# sed: stream editor (non-interactive text editor)
# sed option -i: in-place (output file=input file)
# sed option -n: output only if told via p command
# sed option -E: use extended regular expressions
#
# only lines with "tankstelle_details" or "<sup>"
sed -i -n -E '/tankstelle_details|<sup>/p' tmp
#
# extract the average price from first line of tmp
meanprice=$(sed -n -E \
's/.*text">([0-9])\.([0-9]{2})<sup>([0-9]).*/\1,\2\3/p' \
tmp)
#
# remove first line of tmp to simplify analysis
sed -i '1d' tmp # 1 = first line, d = delete
#
# caveat: not working gas stations at end of tmp
# remove last lines if they contain "tankstelle"
while sed -i '${/tankstelle/!{q1}}' tmp ; do
sed -i '$ d' tmp # delete (d) last line ($) of tmp
```

```
done # loop removing not working gas stations ends
#
date +'%A, %d. %m. %Y, %H Uhr %M'
echo "Webseite 'clever-tanken.de', Treibstoff $t,"
echo "Postleitzahl $plz mit dem Radius $radius km"
#
while [ -s "tmp" ] # loop runs until file is empty
do
line1=$(sed -n '1p' tmp) ; line2=$(sed -n '2p' tmp)
number=$(echo $line1 | sed "s/[^']*'\([^']\{1,\}\)\
'.*/\1/")
station=$(echo $line1 | sed "s/[^']*'[^']\{1,\}'[^\
']*'[^']\{1,\}'[^']*'\([^']\{1,\}\)'.*/\1/")
address=$(echo $line1 | sed "s/[^']*'[^']\{1,\}'[^\
']*'[^']\{1,\}'[^']*'[^']\{1,\}'[^']*'\([^']\{1,\}\
\)'.*/\1/")
price=$(echo $line2 | sed "s/\([0-9]\{1\}\.[0-9]\{\
2\}\)<sup>\([0-9]\).*/\1\2/")
echo "${price} €/l bei ${station} (${address})"
sed -i '1,2 d' tmp # delete lines 1 and 2 of tmp
done
echo "Der mittlere Preis beträgt $meanprice €/l."
```

Die Zeilen 5 bis 7 des Scripts können vom Nutzer geändert werden. Insbesondere wird man den Ort, an dem sich das Auto befindet, in Zeile 5 anpassen. Falls sich dieser Ort öfter ändert, kann man das Programm so ergänzen, dass dieser Ort vom Nutzer abgefragt wird. Der bevorzugte Treibstoff wird in Zeile 6 vorgegeben. In Zeile 7 wird ein Radius festgelegt, der den Umkreis bestimmt, in welchem Tankstellen gesucht werden sollen.

Als Ergebnis erhält man eine Liste mit geöffneten Tankstellen und dem aktuellen Preis des Treibstoffs. Außerdem wird der mittlere Preis der Tankstellen im Umkreis genannt.

Es ist übrigens möglich, bei der Webseite `https://www.clever-tanken.de/` den Ort durch seine geographische Breite (englisch: latitude) und seine geographische Länge zu bezeichnen, statt durch seine Postleitzahl. Der geographische Ort des eigenen Standortsd kann auch automatisch bestimmt werden, siehe oben (Seite 152). Damit ist es möglich, obiges Programm so zu ändern, dass auf die Eingabe einer Postleitzahl verzichtet wird. Allerdings ist die automatische Bestimmung des eigenen Standorts ungenau.

Ein- und Ausgabe mit dialog

Viele Menschen lesen nur noch kurze Texte, die von einem Rahmen umgeben sind. Solch ein Textkasten kann von einem Handy gebildet werden oder durch das „Fenster“ der graphischen Benutzeroberfläche eines Rechners. Nach Installation des Pakets `dialog` kann man auch in einem Shellscript einen Textkasten (englisch: box) erzeugen, in dem Information ausgegeben oder eine Eingabe vom Nutzer abgefragt wird. Es gibt verschiedene Arten von `dialog`-Kästen. Unter anderem:

infobox Textausgabe (Shellscript läuft weiter)
msgbox Textausgabe mit Bestätigung (Script wartet)
fselect Auswahl einer Datei über Pfeiltasten und Leertaste
menu Anzeige eines Menüs mit Auswahl über Pfeiltasten

Im Beispiel für eine `infobox`

```
dialog --infobox "Hallo Welt!" 0 0
sleep 5 # 5 Sekunden Pause
clear
```

wird die Zeichenkette `Hallo Welt!` im Kasten ausgegeben. Die Angaben `0 0` für Höhe und Breite bewirken eine automatische Bestimmung der Fenstergröße. Der Befehl `sleep 5` hält die Programmausführung an, damit der Text gelesen werden kann. Mit `clear` wird der Bildschirm gelöscht.

Bei einer `msgbox`, zum Beispiel

```
dialog --title "Überschrift" \
--msgbox "Bestätigung erforderlich!" 0 0
clear
```

wird ebenfalls eine Zeichenkette (hier: `Bestätigung erforderlich!`) ausgegeben, aber das Script wird erst fortgesetzt, wenn die RETURN-Taste gedrückt wurde. Die Option `title` schreibt einen Titel (hier: `Bestätigung erforderlich!`) über den Kasten. Der Backslash am Ende einer Dateizeile bedeutet, dass der Befehl ununterbrochen in der nächsten Dateizeile fortgesetzt wird.

Nun ein Beispiel zu `fselect`.

```
#!/usr/bin/env bash
# Example for dialog fselect (AHG, 2021)
dir="/home/ahg/video/"
cmd=(dialog --title " Move with arrow keys. Select \
with space-bar. " --fselect $dir 14 80)
vid=$("${cmd[@]}" 2>&1 >/dev/tty)
if [ -z "$vid" ] ; then echo "nothing selected" ; fi
if [ -d "$vid" ] ; then echo "directory selected" ; fi
if [ -f "$vid" ] ; then vlc $vid ; fi
```

Im Auswahlbereich kann man sich mit den Tabulator- und Pfeiltasten bewegen. Die Auswahl geschieht durch Drücken der Leertaste und man bestätigt mit der RETURN-Taste. Im Code legen die Zahlen `14` und `80` Höhe und Breite des Auswahlbereichs fest. Die Bedeutung der if-Verzweigungen wird unten (ab Seite 139) erklärt. Vorausgesetzt wurde hier, dass der Mediaplayer VLC installiert ist und dessen Konfigurationsdatei angelegt wurde, wie unten beschreiben (siehe Seite 195).

Zum Schluss (*so long, king kong*) noch ein Beispiel für ein `menu`.

```
#!/usr/bin/env bash
# Example for dialog menu (AHG, 2021)
fn_a() { echo "see you later, alligator" ; }
fn_b() { echo "after 'while crocodile" ; }
fn_c() { echo "see you soon, racoon" ; }
cmd=(dialog --title " bye bye, butterfly " \
--menu "give a hug, lady bug" 10 60 3)
options=(1 "call fn_a" 2 "call fn_b" 3 "call fn_c")
x=$("${cmd[@]}" "${options[@]}" 2>&1 >/dev/tty)
clear
case $x in 1) fn_a ;; 2) fn_b ;; 3) fn_c ;; esac
```

In der Zeile `cmd=...` wird ein array `cmd` für den ersten Teil des Befehls `dialog` gebildet, aber der Befehl noch nicht ausgeführt. Die Zahlen `10` und `60` legen Höhe und Breite fest, `3` ist die Zahl der gleichzeitig gezeigten Menüeinträge. Das array `options` enthält den zweiten Teil des Befehls `dialog`, die Menüeinträge. In der Zeile `x=...` werden die beiden Teile des Befehls `dialog` zusammengesetzt und der Befehl ausgeführt. Da `dialog` die Ausgabe nach `stderr` (2) sendet, wird sie nach `stdout` (1) umgeleitet und dann der Variablen `x` zugewiesen. Abhängig vom Rückgabewert von `dialog`, also dem Wert von `x` (1, 2 oder 3), wird eine vorher definierte Funktion (`fn_a`, `fn_b` oder `fn_c`) aufgerufen.

Im Verzeichnis `/usr/share/doc/dialog/examples` findet man weitere Beispiele.

Das Ausehen der `dialog`-Kästen kann man in einer Konfigurationsdatei anpassen. Mit

```
dialog --create-rc ~/.dialogrc
```

erzeugt man eine Konfigurationsdatei für den aktuellen Nutzer, die man mit einem Texteditor wie `nano` (siehe Seite 131) bearbeiten kann. Sie enthält bereits Hilfen für die Anpassung und man kann die vorgegebenen Standardwerte ändern. Ich mag es so:

```
use_shadow = OFF
screen_color = (CYAN,BLACK,ON)
```

Das ergibt einen schwarzen Hintergrund ohne Schattenseiten.

5 Games

5.1 ASCII-Art and Telnet

Console on fire

After installing the package `libaa-bin`, the command

```
aafire
```

lights a black and white fire in the console, which is ended with the `Q` key. You can improve the resolution as described below (on page 159).

Screensaver with cmatrix

After installing the package `cmatrix`, the command

```
cmatrix
```

creates a green rain of letters like in the science fiction film „The Matrix“ (Matrix). You can use the # key to change the color to yellow. The `q` key ends the program.

To get the output in a different color, type

```
cmatrix -C yellow
```

for yellow one or correspondingly `red`, `blue`, `white`, `cyan`, `magenta`.

Screensaver with cbonsai

After installing the package `cbonsai` with

```
sudo apt install cbonsai
```

the command

```
cbonsai -liL 42
```

continuously causes bonsai trees to grow on the screen (see figure 5.1) until the program is canceled with the `q` key. The number (here: 42) determines the plant size and should be adjusted to the screen size. The command `cbonsai -h` summarizes the options.

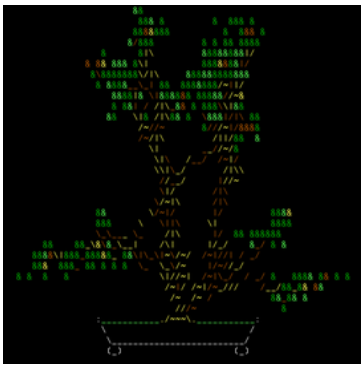


Abbildung 5.1: cbonsai

Steam Locomotive sl

The package `sl` contains a steam locomotive. After

```
sl
```

it rolls across the screen once. Figure 5.2 shows an inverted image (to make it easier to print on white paper). With the option `-l` (English: little) a smaller locomotive runs, with `-a` (accident) you call for help (*HELP*), and with `-F` the locomotive moves up.

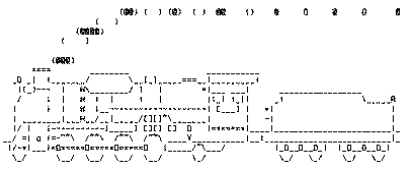


Abbildung 5.2: sl

The command `man sl | less` prints a description of the program.

Video presentation with ASCII-Art: bb

A small movie with black and white ASCII art is available after installing the `bb` package. It is more impressive with better resolution.Therefore, we change the font of the characters (see section 2.1.2 on page 47) to small square characters. After the command

```
sudo dpkg-reconfigure console-setup
```

we make the following settings in the menu:

```
UTF-8
Guess optimal character set / Vermutlich optimaler Zeichensatz
VGA
8 x 8
```

The movie starts after the command

```
bb
```

It can be ended by pressing the `Q` key. In order to then work with the usual font in the terminal, set a larger font again, as described in section 2.1.2 on page 47.

Big words with figlet or toilet

After installing the `figlet` package, the command

```
figlet -c -W "Hallo Welt!"
```

writes the specified text to the screen in capital letters formed by dashes. Figure 5.3 shows an inverted image to make it easier to print on white paper.

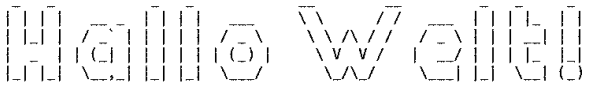


Abbildung 5.3: figlet

The `-c` option centers the output and `-W` provides spacing between characters. The command `man figlet | less` prints a description of the program.
Furthermore, there is a Python package `pyfiglet` for the use in Python programs, which can be installed with the command `sudo apt install python3-pyfiglet`

An alternative is the `toilet` package. It is installed using the command

```
sudo apt install toilet
```

and displays the specified text colorfully with the command

```
toilet -F gay "Hallo Welt!"
```

Say it with cows: cowsay

After installing the `cowsay` package, the command

```
cowsay "Kuh-ten Tag!"
```

produces a cow formed from characters (ASCII) with a speech bubble showing the specified text. The text can also come from a file or, in a shell script, from a variable. Figure 5.4 shows an inverted image to make it easier to print on white paper.

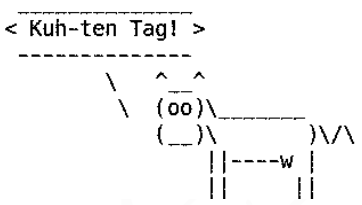


Abbildung 5.4: cowsay

There are variations of the image, for example shows

```
cowsay -f turtle "This may take a while."
```

a turtle with a speech bubble. The command `cowsay -l` outputs a list of available images. By the way: using `cowthink` instead of `cowsay` turns the speech bubble into a thought bubble.
Since the created image consists of characters, you can save it as a text file:

```
cowsay -f elephant "Please Ignore Me" > cowfile
```

You can also save the text file as an image (see page 174). A description of the program `cowsay` can be obtained with the command `man cowsay | less`

If you have installed `fortune-mod`, `fortunes-min`, `fortunes` and `librecode0`, the cow can also quote clever sayings. The command for this is

```
fortune | cowsay
```

For German sayings you also have to install the `fortunes-de` package. If the English language was selected in the operating system, the command

```
fortune /usr/share/games/fortunes/de | cowsay
```

will output German sayings.

Say it with dinosaurs: dinosay

The package `dinosay` is installed by

```
pip3 install --user dinosay --break-system-packages
```

`dinosay` creates a dinosaur with a speech bubble. You have to specify the dinosaur species with an option, for example with the `-d stego` option.

You can (but don't have to) choose a color, for example with the `-c yellow` option. The text following the options must be enclosed in quotation marks. Figure 5.5 shows an example with a Brachiosaurus.

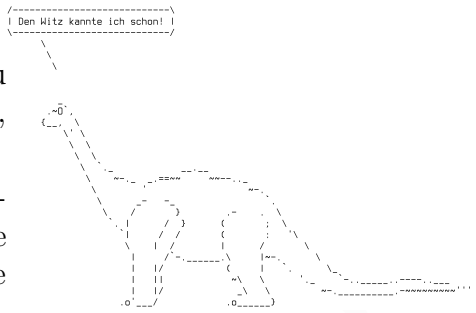


Abbildung 5.5: dinosay

```
dinosay -d brachio -c green "This joke is older than me!"
```

The package also provides a Python module. An example of this:

```
1 from dinosay import dinoprint, DINO_TYPE
2 msg = "I'm not a groundhog!"
3 dinoprint(msg, DINO_TYPE["triceratops"])
```

Text in a beautiful frame: boxes

The `boxes` package makes it possible to draw simple or pictorial boxes around text, for example with the command

```
echo "Happy Birthday, Frankenstein!" | -d girl -a c -p a2t1
```

Figure 5.6 shows an inverted image to make it easier to print on white paper. The string sent by the command to standard output is redirected with a pipe (`|`) to the program `boxes`, which produces the framed output. The option `-d girl` selects the design of the frame, `-a c` positions the text in the frame (here: centered) and `-p a2t1` determines the distances between the text and the surrounding frame (here: 2 on all sides, except top 1). The command `boxes -l | less` shows the different designs available and `man boxes | less` prints a description of the program.

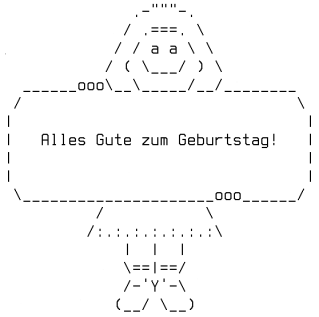


Abbildung 5.6: boxes

Image to ASCII: jp2a

The program `jp2a`, installed by the package of the same name, converts an image file in JPG or PNG format into text that displays the same image on the screen, but in poor resolution. The option `--color` yields colored characters. If you like the result, you can take a screenshot using the `fbgrab` program (see page 225).

Timer with colorful cat: nyancat

Sometimes you need a timer that shows the seconds that have passed. For cat lovers there is the timer `nyancat`, which is installed via git. If `git` has not yet been installed, you can do so with

```
sudo apt install git
```

With `git` you can install *nyan cat* using the following commands:

```
git clone https://github.com/klange/nyancat.git
cd nyancat
make
sudo make install
```

You start the timer with the command

```
nyancat
```

and end it with the keyboard shortcut `Ctrl-C` (see page 12).

Telnet services

Telnet is an old system for character-oriented data exchange on the Internet. On our computer we can use the command line to call a program (Telnet client) that establishes a connection to a remote computer and receives data from another program (Telnet server). After installing with the command

```
sudo apt install telnet
```

various services are available that work with text files or show Ascii art. Data exchange via Telnet is not secure and should therefore be limited to services where eavesdropping is not dangerous. A running Telnet service can be aborted using the key combination `Ctrl-]` (pressing the `Ctrl` key and the `]` key at the same time), and then you can exit the `telnet` program with the command `quit`.

The command `telnet X` starts Telnet and opens a connection to a server `X`. With

```
telnet
```

you simply start a telnet session and then the command `?` lists possible Telnet commands. For instance, the command `open X` opens a connection to a server `X`.

Maps with mapscii

You can view the OpenStreetMap maps with Ascii Art via Telnet. To make the resolution somewhat usable, we change the font of the characters (see section 2.1.2 on page 47) to small square characters. After the command

```
sudo dpkg-reconfigure console-setup
```

you make the following settings in the menu:

```
UTF-8
Guess optimal character set / Vermutlich optimaler Zeichensatz
VGA
8 x 8
```

The map viewer mapscii can now be started with the command

```
telnet mapscii.me
```

Because of the poor resolution and the small number of colors (6), you shouldn't expect a magnificent spectacle. Figure 5.7 shows Europe.

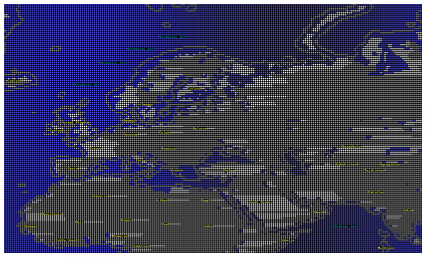


Abbildung 5.7: mapscii

The following keys control the map:

↑ ↓ ← → moving the map
a reduce scale (zoom in)
z increase scale (zoom out)
q stopping the mapscii service

To restore the usual font size in the terminal, set the font as described in section 2.1.2.

An ASCII-Art Star Wars movie

A Starwars movie, displayed with ASCII art, runs in the console after to the command

```
telnet towel.blinkenlights.nl
```

The movie can be canceled with the keyboard shortcut `Ctrl-]`. You can then exit Telnet with the command `quit`

Telehack

A connection with Telehack takes us into the past of the Internet.

```
telnet telehack.com
```

An input is expected after a point. Among other things, the following offers are available:

? shows the list of available commands (except `quit`)
geoip ***ipa*** determines the geographical location of a passed IP address *ipa*
ipaddr shows our IP address and our internet service provider
joke delivers a clever saying or joke (in English)
morse ***text*** translates text *text* into Morse code, or vice versa
zork opens the game ZORK, an old text adventure (≈ 1980), `Ctrl-C` closes it

Telehack is case-insensitive. You can leave Telehack by typing `quit`.

Timer nyancat via Telnet

The timer `nyancat` presented on page 162 can be used via Telnet with the command

```
telnet nyancat.acc.umu.se
```

Then `nyancat` does not have to be installed on your own computer. The timer is ended with the key combination `Ctrl-]`. You can then exit Telnet with the command `q` or `quit`. The `reset` command returns the terminal to its normal state (black background instead of blue).

5.2 Console games

Anyone who asks what the point of games with poor graphics is, should skip this chapter. Now, having separated the chaff from the wheat, let's look at some games in alphabetical order. The risk of addiction is low. An (albeit incomplete) overview of Debian's console games can be found at `https://blends.debian.org/games/tasks/console.en.html`.

5.2.1 Adventures and brain games

Angband

Angband is an old computer role-playing game that is being further developed. You install it with

```
sudo apt install angband
```

and you start it with the command

```
angband -mgcu
```

A character wanders through the fantasy world of the novel The Lord of the Rings by J. R.R. Tolkien. The graphics are simple, but the game is extensive and the operation requires learning. Just study The Angband Manual. To exit the game, press the key combination `Ctrl-X` and the Escape key.

There are many game variations and a forum of fans.

Colossal Cave Adventure

The `bsdgames` package contains, among other things, the text-based interactive adventure game *Colossal Cave Adventure*, or *Advent* for short. You start it with the command

```
adventure
```

The game was created in 1976–1977 and greatly influenced the development of computer games. It is therefore a must for every tradition-conscious nerd!

The player explores a cave where numerous dangers lurk („A huge green fierce snake bars the way!“). It is controlled by entering simple text commands that consist of one or

two short English words, for example `go west` or `look` or `get lamp`. Since the program analyzes only the first five letters of a word, a cardinal direction such as *northwest* must be abbreviated as `nw` to distinguish it from *north*. Sometimes magic is necessary: *„xyzzy“*. The goal is to collect treasures and points and then leave the cave alive. It is useful to draw a map of the cave and take notes.

Dungeon Crawl

After installing the package `crawl`

```
sudo apt install crawl
```

the adventure role-playing game *Dungeon Crawl* is available in version *Stone Soup*. The game idea is similar to that of the old computer game *Rogue* (released in 1980).

The game is started with the command

```
crawl
```

A *Tutorial for Dungeon Crawl* explains the complex game. To exit, use the key combination `Ctrl-q` followed by `yes`.

Freesweep

After installing the package `freesweep`

```
sudo apt install freesweep
```

the mine clearance game (*minesweeper*) is started with

```
freesweep
```

In the queries at the beginning you can use ENTER to accept the standard settings. You move around the board using the arrow keys. The space bar opens a field. If you hit a mine, you lose. Otherwise a number indicates the number of mines on the neighboring fields. If you suspect you know a minefield, mark it with `f`. `?` shows all possible key commands. The aim of the game is to mark all the mines and open the other fields.

Gnuchess

After installing the package `gnuchess` and starting the program with

```
gnuchess
```

you can play chess against the computer. The moves are entered in English chess notation, for example `Nc3` to place the knight on the c3 square. The „graphical representation“ of the chessboard contains letters for the pieces and dots for empty fields. The program can be ended by entering `quit` or with the keyboard shortcut `Ctrl-C` (see page 12).

The command `gnuchess --version` shows the installed version and the command `info gnuchess` prints a game guide.

If you call `gnuchess` in the terminal `fbterm` (see section 3.2.3 on page 86) with the command

```
gnuchess -g
```

the chess pieces are represented by special characters, the Unicode chess symbols (code points U+2654 to U+265F), see figure 5.8.

`-g` is the short form of `--graphic`. Whether the display is attractive depends on the font used by `fbterm`.

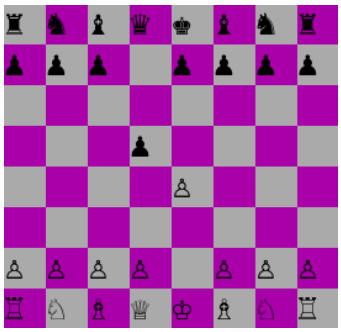


Abbildung 5.8: gnuchess

GNU Go

The package gnugo with the strategic board game Go or Weiqi is installed with

```
sudo apt install gnugo
```

and starts with the command

```
gnugo --color black
```

where you play with the black stones and the computer (the opponent) plays with the white ones. In the above command you can of course also choose `white`.

You place a stone by entering the x coordinate (`A` – `T`) and the y coordinate (`1` – `19`), for example `Q16`. Each entry is completed by pressing the ENTER key. By entering `quit` you leave the game.

Gomoku

The strategic board game *gomoku* (Five in a Row) is included in the package `bsdgames`. The game starts with the command

```
gomoku
```

You play against the computer, black against white. The fields on a 19 ×19 game board are addressed using coordinates.

Black starts. Whoever's turn it is places a stone. The goal is to place five stones in a row. Figure 5.9 shows an inverted image to make it easier to print on white paper.

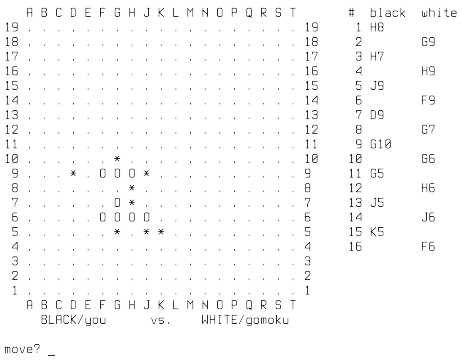


Abbildung 5.9: gomoku

Greed

The task in the game `greed` is to remove („eat away“ as many as possible from a given colored number field. You choose the direction of movement of a cursor (@) using the arrow keys, the keys on the numeric keypad or the keys `h`, textttj, `k`, `l`, `y`, `u`, `b` and `n`. The first digit in the direction of movement is the route length. The game ends when there are no more legal moves. With the command

```
sudo apt install greed
```

the program is installed. The `?` key outputs help text, and the `p` key switches the display of possible moves on and off. Press the `q` key followed by `y` to end the program. You can get more information using the command `man greed`.

HoiChess

In addition to GNU Chess (see above, page 165), HoiChess is a chess program that can run in the console. After installing the `hoichess` package and starting the program with the `hoichess` command, you can play chess against the computer, similar to GNU Chess. The program is ended by entering `quit`.

Klondike (solitaire)

Klondike is a card game for a single player (solitaire). It is installed using the command

```
sudo apt install tty-solitaire
```

and it starts with the command

```
ttysolitaire --no-background-color
```

The rules of the game are assumed to be known, but the control keys are explained initially. The `q` key ends the program.

Nudoku

Nudoku creates a Sudoku, a Japanese number puzzle. After installing the package `nudoku` it is called with the command

```
nudoku -d normal
```

The option `-d` allows you to choose the difficulty. Instead of `normal`, `easy` and `hard` are also possible. You can move around the playing field using the cursor keys. The functions of other keys are explained on the game board. Use `Q` or the key combination `Ctrl-C` to exit the program (see page 12).

Robots

The strategic board game *Robots* is also included in the `bsdgames` package. The command

```
robots
```

starts the game. The player (`@`) is pursued by robots (`+`) and has to avoid them. He can destroy robots by causing them to collide. They then become a pile of scrap (`*`) that other robots can smash into. If necessary, the player can teleport. The button controls are described next to the game board.

Chess with chs

In the terminal `fbterm` (see page 86) you can draw a chessboard with `chs`. This can be used to play against a chess engine like Stockfish, see

`https://itsfoss.com/play-chess-linux-terminal/`

If you want a larger chessboard, you can start `fbterm` (before calling `chs`) with an option to increase the font size, for example like this:

```
fbterm --font-size=48
```

Tic-Tac-Toe and Reversi

AP Games offers us the board games tic-tac-toe (noughts and crosses) and Reversi (Othello) as a Python program with artificial intelligence. Installation is done with

```
pip3 install --user ap-games --break-system-packages
```

and a game of tic-tac-toe with difficulty `easy` starts with the command

```
python3 -m ap_games 1 user easy
```

Higher levels of difficulty are `medium`, `hard` and `nightmare`. In the order `user easy` you have the first move, in the reverse order the computer starts.

If you want to play Othello, enter a `2` instead of `1`. You move by entering the x and y coordinates, separated by a space, and pressing the Enter key. You can cancel the game with `Ctrl-C`.

2048

The game 2048 is installed with the command

```
sudo apt install 2048
```

and starts with the simple command

```
2048
```

showing a 4×4 array, whose elements (tiles) are either empty or contain a number 2^n, with $1 \leq n^{10}$, on a colored background. The array starts with fourteen empty tiles and two randomly placed tiles that contain $2^1 = 2$.

The array is then changed using the four arrow keys, resulting in tiles with larger numbers. Each move slides all tiles in the same direction and touching tiles containing the same number merge into a pair of tiles, each with the added value of the merged ones. A move that pushes two tiles with 1024 together, yielding 2028, successfully ends the game.

5.2.2 Games of skill

ballerburg

ballerburg is described as „a classical castle combat game“, whatever that means. It is installed with the command

```
sudo apt install ballerburg
```

and, using the Browser `w3m` (see page 209), you can read the manual with

```
w3m /usr/share/doc/ballerburg/manual.html
```

in English, or with

```
w3m /usr/share/doc/ballerburg/anleitung.html
```

in German. You can compete against the computer or against a second player. However, you need a mouse to play the game (see page 48). The game is cancelled with the `q` key.

Bastet

Bastet is a Tetris-type game in which falling blocks are quickly to be placed in the right place. After installing the package,

```
sudo apt install bastet
```

it is called with the command

```
bastet
```

Figure 5.10 shows an inverted image to make it easier to print on white paper. The blocks are moved using the arrow keys and they are rotated using the space bar. The program can be aborted using the key combination `Ctrl-C` (see page 12).

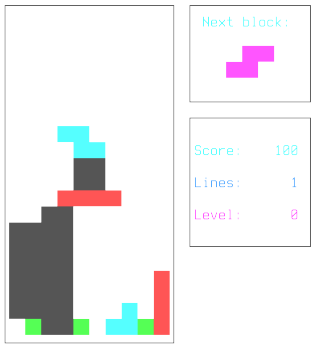


Abbildung 5.10: bastet

Block Attack - Rise of the Blocks

Block Attack is another colorful Tetris-type game. It is installed with

```
sudo apt install blockattack
```

and works well with keyboard or mouse (if `gpm` is installed and configured, see page 48).

Moon Buggy

moon-buggy is limited to the essential gaming experience. It is installed with

```
sudo apt install moon-buggy
```

and started by

```
moon-buggy
```

The image is composed of letters (ASCII-Art). Options of the very very retro game are listed by the `moon-buggy -h` command. Like an alcoholic astronaut, you start with the space bar. In the game you jump up with the `j` key or (same) with the spacebar, and you shoot with the `a` or `l` key. You can cancel the game with the `q` key.

nInvaders

You can simply shoot down evil aliens in the game nInvaders, which is installed with

```
sudo apt install ninvaders
```

and started with

```
ninvaders
```

You move the cannon with the left and right arrow keys and shoot with the space bar. You can cancel the game with the `q` key. The command `man ninvaders` tells more and `ninvaders -h` explains how to specify a level of difficulty.

Nsnake

A wandering snake can be fed after installing package `nsnake` using command

```
sudo apt-get install nsnake
```

and starting it with

```
nsnake
```

The snake is controlled with the arrow keys. The `p` key starts and ends a break, `h` shows help and `q` ends the game. To use different levels, copy the corresponding files with

```
cp /usr/share/games/nsnake/levels/* $HOME/.local/share/nsnake/levels/
```

once into a subdirectory of your home directory.

Pacman4console

After installing package `pacman4console`

```
sudo apt install pacman4console
```

the game is started with the command

```
pacman4console
```

You can wander around a two-dimensional maze as Pac-Man, see figure 5.11, controlled with arrow keys.

Pac-Man has to collect all the dots (*Pac-Dots*) in the maze to reach the next level. He has to avoid the four ghosts (Blinky, Pinky, Inky and Clyde), because touching them costs a life. When all lives are used up, the game is over. If Pac-Man encounters a strengthening *Power Pellet*, he can destroy the ghosts for a short time (in the first levels) by touching them. Of course that gives you extra points. Sometimes fruits appear; they also give extra points.

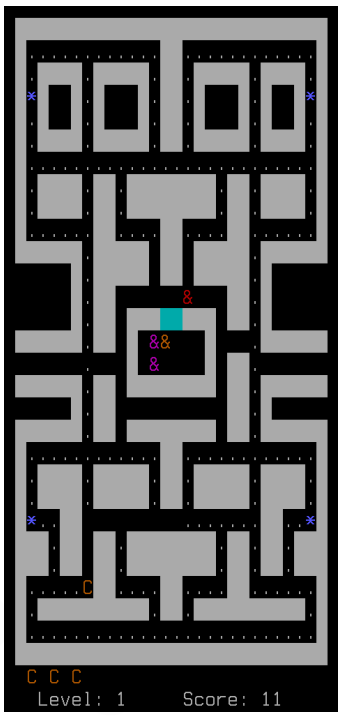


Abbildung 5.11: pacman

6 Multimedia und Internet

6.1 Bilder, PDF, Audio und Video

Video-, Audio- und Bilddateien können in einem *Containerformat* gespeichert werden, sodass verschiedene Datenströme (Video, verschiedene Audiospuren, Untertitel, Bilder, Begleittext) in einer Datei zusammengefasst werden. Die Datei enthält Information zu ihrem Inhalt als *Kopfdaten*. Bei manchen Containerformaten ist die Codierung der Daten festgelegt, andere geben die gewählte Codierung in den Kopfdaten an.

Für das Abspielen von Audiodateien oder Internetradio gibt mehrere geeignete Programme. Neben den universellen Mediaplayern `mpv`, `MPlayer` und `VLC` gibt es auch `MOC`. Eine Auswahl von Internet-Radiosendern findet man im Anhang (Seite 412).

Das Abspielen von Videodateien oder Internetfernsehen ist auch den universellen Mediaplayern `mpv`, `MPlayer` und `VLC` möglich. Da wir auf die Unterstützung durch fensterorientierte display server (wie X oder Wayland) verzichten, müssen wir Programme einsetzen, die Graphik mit dem framebuffer auf dem Monitor darstellen können. Dabei ist zu beachten , dass Datenströme für Video den Rechner umso mehr belasten, je höher die Bildauflösung ist, siehe Tabelle 6.1.

Name	horizontal	vertikal	Verhältnis	Anzahl Pixel
1/8 VGA	240	180	4 : 3	43 200
240p	320	180	16 : 9	57 600
Quarter VGA, QVGA	320	240	4 : 3	76 800
Half VGA, HVGA	480	320	3 : 2	153 600
360p	640	360	16 : 9	230 400
VGA	640	480	4 : 3	307 200
480p	854	480	16 : 9	409 920
HDTV, 720p	1280	720	16 : 9	921 600
Full HD, 1080p	1920	1080	16 : 9	2 073 600

Tabelle 6.1: Einige übliche Bildauflösungen für Video-Datenströme

`mpv` und `VLC` können zur Decodierung des Video-Datenstroms die „Hardwarebeschleunigung“ (hardware acceleration) der GPU des Raspberry Pi benutzen und damit Videos

effizient darstellen. Die Einstellung des VLC über die Konsole ist aber manchmal schwierig, da VLC hauptsächlich für fensterorientierte display server entwickelt wird.

Der früher eingesetzte Mediaplayer mit „Hardwarebeschleunigung“, `omxplayer`, wird in neuen Betriebssystemen leider nicht mehr unterstützt (im Raspberry Pi OS seit der Version vom 30. Oktober 2021). Er gilt als veraltet und wird nicht weiterentwickelt.

Der bewährte `MPlayer` kann auch Videos wiedergeben. Nachteilig ist die fehlende „Hardwarebeschleunigung“. Für alle Mediaplayer ist die Verwendung von playlists zu empfehlen, die man selbst erstellen kann (siehe Seite 197) oder im Internet finden kann.

6.1.1 Bildanzeige (fbi, fim), Textdatei als Bild, animierte GIFs

Bilddateien können mit dem Framebuffer (Bildspeicher) des Betriebssystems dargestellt werden. Das kann mit dem Programm `fbi` (erfordert Installation des Pakets `fbi`) oder dem Programm `fim` (nach Installation von `fim`) geschehen, die wir in diesem Abschnitt besprechen. Sie unterstützen die Bildformate BMP, GIF (ohne Animation), JPEG, PNG, PPM und TIFF, .xwd. Außerdem gehen wir in diesem Abschnitt auf die Wiedergabe von animierten Gif-Bilder ein.

Später (ab Seite 387) sehen wir, wie man Bilddateien mit `Pygame` darstellt.

fbi

Eine Datei `/home/ahg/images/bild1.jpg` wird mit dem Befehl

```
fbi /home/ahg/images/bild1.jpg
```

gezeigt. Mit der Option `--noverbose` wird die Statuszeile weggelassen.

Eine Steuerung mit folgenden Tasten möglich:

+	Bild vergrößern
–	Bild verkleinern
a	Vergrößerung auf Bildschirmgröße
r	Bild um 90° im Uhrzeigersinn drehen
h	Hilfe (Anzeige der Tastaturbefehle)
q	(quit) Programm beenden

fim

Eine Datei `/home/ahg/images/bild2.png` wird mit dem Befehl

```
fim -a /home/ahg/images/bild2.png
```

bildschirmfüllend gezeigt. Wenn man die Die Option `-a` weglässt, wird das Bild in seiner natürlichen Größe dargestellt. Mit der Option `-q` wird die Statuszeile weggelassen. Durch

+	Bild vergrößern
–	Bild verkleinern
f	oben/unten vertauschen (spiegeln)

m links/rechts vertauschen (spiegeln)
q (quit) Programm beenden

kann man das Programm über die Tastatur steuern.

Textdatei als Bild speichern

Für das Programm `pango-view` brauchen wir das Paket `pango1.0-tools`. Es wird mit

```
sudo apt install pango1.0-tools
```

installiert. Nun können wir eine Textdatei als Bild speichern. Wenn wir eine Textdatei mit ASCII-Art, zum Beispiel die auf Seite 160 beschriebene Datei `cowfile`, in ein Bild mit dem Format PNG umwandeln wollen, dann sollte eine nichtproportionale Schriftart (`mono`) gewählt werden und die Bildauflösung in der Einheit dpi ausreichend groß sein. Dies gelingt mit dem Befehl

```
pango-view --dpi=300 --font="mono" -qo cow.png cowfile
```

und die Bilddatei heißt hier `cow.png`. Sie kann mit `fbi` (siehe Seite 173) oder `fim` (siehe Seite 173) auf dem Bildschirm dargestellt werden.

Auf dem Umweg über eine Bilddatei kann man auch Textdateien darstellen, die Schriftzeichen enthalten, welche die Konsole nicht darstellen kann. Zum Beispiel kann man einen chinesischen Text anzeigen, vorausgesetzt ein entsprechender chinesischer Font wurde installiert (siehe Seite 79). Die Option `--font="mono"` wird dann in obigem Befehl meistens nicht benötigt.

Wiedergabe von animierten GIFs

Animierte Gif-Bilder kann der Mediaplayer `mplayer` wiedergeben, siehe Seite 190.

Man kann animierte GIFs auch als ASCII-Art wiedergeben, wenn reduktionistische Verfremdung wichtiger ist als der eigentliche Bildinhalt. Auch das kann grundsätzlich der `mplayer`, ist aber manchmal damit überfordert (technisch, nicht künstlerisch). Hier kann das Programm textttgif-for-cli helfen.

Zunächst installiert man einige Bibliotheken mit dem Befehl

```
sudo apt install ffmpeg zlib* libjpeg* python3-setuptools
```

und ignoriert die Warnungen. Dann folgt der Befehl

```
pip3 install --user gif-for-cli
```

und nun kann man animierte GIFs abspielen. Für eine bessere Auflösung sollte der Font der Schriftzeichen (siehe Abschnitt 2.1.2 auf Seite 47) auf kleine quadratische Zeichen umgestellt werden, wie auf Seite 159 beschrieben wird. Dann wird zum Beispiel eine GIF-Datei `lustig.gif` mit dem Befehl

```
gif-for-cli lustig.gif
```

als Animation gezeigt. Mit der Option `-l` erhält man eine Endlos-Schleife, die mit `Ctrl-C` abgebrochen werden kann. Um danach mit der gewohnten Schriftart im Terminal zu arbeiten, stellt man anschließend wieder einen größeren Font ein, wie in Abschnitt 2.1.2 auf Seite 47 beschrieben.

6.1.2 Bildaufnahme mit einer Kamera

Bildaufnahme mit einer Pi Camera

Wenn eine Pi Camera angeschlossen wurde (siehe Seite 18) kann man sie mit dem `libcamera` Treiber steuern. Für einen einfachen Test genügt der Befehl

```
libcamera-hello
```

Damit sollte für etwa fünf Sekunden ein laufendes Kamerabild (Vorschau) gezeigt werden. Zu unterscheiden sind die Größe (Breite und Höhe) bei der Speicherung eines Kamerabilds und die Größe der Vorschau (preview). Mit

```
libcamera-jpeg -o bild.jpg
```

wird das laufende Kamerabild als Vorschau fünf Sekunden gezeigt und dann ein Bild in hoher Auflösung (abhängig vom Kameratyp, zum Beispiel 3280 pixels × 2464 pixels) aufgenommen und in der Datei `bild.jpg` im Format JPEG gespeichert. Das Bild kann anschließend mit dem Programm `fbi` oder `fim` betrachtet werden (siehe Abschnitt 6.1.1 auf Seite 173). Optionen sind unter anderem

--help ohne weitere Optionen und ohne Dateiname: Hilfe (Liste der Optionen)
--version Ausgabe der Versionsnummer des Programms auf dem Bildschirm
--list-cameras Ausgabe der verfügbaren Kameras und der Bildgrößen (sensor modes)
--camera *x* wähle Kamera x, wenn mehrere verfügbar sind (0 = erste Kamera)
--height *x* Höhe des zu speichernden Bilds = x pixels (x ganzzahlig)
--width *x* Breite des zu speichernden Bilds = x pixels (x ganzzahlig)
--hflip horizontale Spiegelung des zu speichernden Bilds
--vflip vertikale Spiegelung des zu speichernden Bilds
--ev *x* Belichtungsänderung x, positiv: heller, negativ: dunkler (zum Beispiel -1.5)
--contrast *x* Kontrast x, Standard: 1, erhöhter Kontrast: `--contrast 1.5`
--brightness *x* Helligkeitsänderung x, Standard: 0, dunkler: `--brightness -0.2`
--saturation *x* Farbsättigung x, Standard: 1, etwas farbiger: `--saturation 1.2`
--sharpness *x* Schärfung x, Standard: 1, mehr: `--sharpness 2.5` (0 no sharpening)
--codec *x* nur `libcamera-vid`: Codec x (`h264`, `mjpeg`, `yuv420`, Standard ist `h264`)
–c *x* benutze Konfigurationsdatei x (siehe Beispiel, unten)
–e *x* Speichern des Bilds im Format x (`jpg`, `png`, `rgb`, `bmp`, `yuv420`, Standard `jpg`)
–n keine Bildausgabe (Vorschau) vor der Aufnahme, die standardmäßig erfolgt
–o *x* Speichern des Bilds in Datei x (Endung sollte zum Format passen)
–p $x, y, \Delta x, \Delta y$ Ort und Größe des Vorschaubilds, zum Beispiel `-p 0,0,320,240`
–q JPEG quality, nur für das Format jpg, 0 – 100 (beste Qualität), Standard ist 93

–t *x* laufendes Bild für x Millisekunden vor der Aufnahme (Standard 5000)
–v Ausgabe von mehr Information auf dem Bildschirm (englisch verbose = wortreich)

Mit dem Befehl

```
libcamera-jpeg -t 1 --hflip --vflip -o "bild_$(date +%y%m%d_%H%M%S).jpg"
```

wird ein Bild fast ohne Verzögerung (1 ms) im Format JPEG aufgenommen und um 180 ° gedreht. Der Name der Datei enthält Datum und Uhrzeit der Aufnahme.

Man kann Optionen in einer Datei festlegen, zum Beispiel `myconf.txt`, und diese beim Aufruf von `libcamera-jpeg` oder `libcamera-vid` in der Option `-c myconf.txt` nennen. Genauer gesagt: Es muss der Pfad zur Datei genannt werden. Die Optionen müssen in der Form `Schlüssel=Wert` eingetragen werden, oder in der Form `Schlüssel=`, wenn es keinen Wert gibt. Schlüssel müssen in der Langform angegeben werden (`timeout` statt `t`) und die Doppelstriche `--` entfallen am Anfang (`hflip=` statt `--hflip`). Mit `#` eingeleitete Kommentare sind möglich. Ein einfaches Beispiel:

```
timeout=3000    # Dauer: 3 Sekunden
hflip=          # horizontal spiegeln
vflip=          # vertikal spiegeln
```

Eine „unendliche“ Videoübertragung von der ersten Kamera bekommt man mit

```
libcamera-vid -t 0
```

bis zum Abbruch des Programms durch `Ctrl-C` (siehe Seite 12). Soll das Bild um 180 ° gedreht werden, verwendet man die Optionen `--hflip` und `--vflip` (siehe oben).

Einen Videofilm ohne Ton im Format Motion JPEG (MJPEG) kann man mit

```
libcamera-vid --width 640 --height 480 -t 5000 --codec mjpeg -o v.mjpeg
```

aufnehmen. Bildbreite × -höhe ist 640 pixels × 480 pixels und die Dauer ist 5 Sekunden (5000 Millisekunden). Die Wiedergabe kann mit dem MPlayer erfolgen (siehe Seite 190).

```
mplayer v.mjpeg
```

Die Option `--codec` ***x*** bestimmt den video codec (siehe Seite 66). Standardwert ist `h264` für den H.264 Codec, auch AVC (Advanced Video Coding) genannt.

Nun soll das laufende Bild der Kamera in ein lokales Netzwerk eingespeist werden, zum Beispiel zur Videoüberwachung unseres Aquariums. Zunächst müssen wir die IP-Adresse des Raspberry Pi wissen. Diese erfahren wir zum Beispiel mit dem Shellscript für System-Information, das ab Seite 146 beschrieben wurde.

Alternativ kann man den Befehl `ifconfig` geben und sich die Antwort auf dem Bildschirm anschauen. Wenn der Raspberry Pi über WLAN an das Netzwerk angeschlossen ist, findet man im Abschnitt, der mit `wlan0:` markiert ist, die WLAN-Adresse zwischen den Worten `inet` und `netmask`.

In folgendem Beispiel nehmen wir an, dass die IP-Adresse des Raspberry Pi im lokalen Netzwerk `192.168.2.173` ist. Außerdem müssen wir einen sogenannten Port als Zahl angeben und nehmen (willkürlich) `8888` dafür. Die Art der Signalübertragung im Netzwerk wird durch ein Netzwerkprotokoll bestimmt, zum Beispiel TCP, UDP oder RTSP (was auf TCP oder UDP aufbaut). Wir verwenden hier TCP. Mit dem Befehl

```
libcamera-vid -t 0 --inline --listen -o tcp://0.0.0.0:8888
```

beginnen wir die Aussendung des Videosignals, die erst mit `Ctrl-C` beendet wird. Nun können wir auf einem anderen Rechner, der ebenfalls im lokalen Netzwerk liegt (also am gleichen Router angeschlossen ist), das Videosignal empfangen. Nehmen wir zum Beispiel einen Linuxrechner mit graphischer Oberfläche und VLC als Mediaplayer, dann wählen wir im VLC „Medien öffnen – Netzwerk“ und tragen unter „Bitte geben Sie eine Netzwerkadresse ein:“ die Zeile

```
tcp/h264://192.168.2.173:8888
```

ein und klicken auf „Wiedergabe“. Mit wenigen Sekunden Verzögerung sollte das Videosignal des Raspberry Pi (Sender) auf dem Linuxrechner (Empfänger) zu sehen sein. Man muss natürlich die richtige IP-Adresse eintragen! Man kann eine andere Portnummer wählen, aber sie muss bei Sender und Empfänger übereinstimmen.

Eine Dokumentation zu Raspberry Pi Cameras gibt bei der Raspberry Pi Foundation auf `https://www.raspberrypi.com/documentation/accessories/camera.html`

Bildaufnahme mit Webcam über USB

Eine oder mehrere USB-Webcams lassen sich leicht anschließen. Nachteilig ist bei Photos die meist geringere räumliche Auflösung gegenüber einer Pi Camera (siehe Seite 175). Bei Videos ist die Auflösung aber oft ausreichend. Für viele USB-Webcams hat das Betriebssystem bereits einen passenden Treiber installiert. Auf der Webseite `https://elinux.org/RPi_USB_Webcams` findet man eine Liste geprüfter Webcams. Zu beachten ist, dass einige Webcams, eventuell abhängig vom Modell des Raspberry Pi, über einen aktiven USB-Hub betrieben werden sollten. Es gibt jedoch Webcams, die problemlos direkt an eine USB Buchse des Raspberry Pi angeschlossen werden können. Oft funktionieren einfache, ältere USB-Webcam besser als neue.

Leider werden manche USB-Webcams vom neuen Reaspberry PI OS (bullseye) nicht mehr als Kamera erkannt (wohl aber als USB-Gerät), die mit der Vorgängerversion des Betriebssystems (buster) funktionieren.

Vor und nach dem Einstecken der USB-Webcam gibt man den Befehl

```
lsusb
```

und wenn beim zweiten Mal eine Zeile mehr ausgegeben wurde, wissen wir, dass das Betriebssystem die USB-Webcam zumindest als Gerät erkannt hat.

Die video codecs (siehe Seite 66) des Treiberprogramms, die räumliche Auflösung (Bildbreite und -höhe, englisch resolution), mit welcher die Kamera Bilder erzeugen kann, und mögliche Bildfrequenzen (frame rates), erfährt man mit

```
v4l2-ctl --list-formats-ext | less
```

Um Bilder oder Videos aufnehmen zu können, braucht man ein entsprechendes Programm. Für einzelne Bilder ist `fswebcam` zu empfehlen, das mit dem gleichnamigen Paket installiert wird (siehe Seite 79). Es benötigt wenig Speicherplatz. Der Befehl

```
fswebcam -r 640x480 test.jpg
```

nimmt ein Bild mit der Auflösung 640 × 480 auf und speichert es in der Datei `test.jpg`. Das Bild kann anschließend mit dem Programm `fbi` oder `fim` betrachtet werden (siehe Abschnitt 6.1.1 auf Seite 173). Die Bildaufnahme kann durch Verwendung von Optionen noch verfeinert werden. Auskunft über die Optionen gibt der Befehl

```
man fswebcam
```

Man kann die gewählten Optionen in einer Konfigurationsdatei, hier /home/ahg/bild.cfg genannt, speichern. Ihr Inhalt könnte zum Beispiel so aussehen:

```
# fswebcam - Konfiguration
device "/dev/video0"
input 0
delay 1
resolution 640x480
set brightness=80%
set contrast=16%
skip 5
frames 1
scale 320x240
banner-colour "#FF000000" # farblos transparenter Bannerhintergrund
line-colour "#FF000000" # farblos transparente Bannerlinie (keine Linie)
text-colour "#000000"
title "Webcam Testbild"
timestamp "%d.%m.%Y %H:%M"
jpeg 90
save "/home/ahg/image/bild.jpg"
```

Um eine Bildaufnahme durchzuführen, gibt man den Befehl

```
fswebcam -c /home/ahg/bild.cfg
```

ein. Die Option -c bestimmt die Verwendung der danach genannten Konfigurationsdatei. Das Bild wird hier in der Datei `/home/ahg/image/bild.jpg` gespeichert. Falls eine Datei gleichen Namens bereits existiert, wird diese überschrieben.

Die Benutzung von `fswebcam` in einem Pythonprogramm ist einfach:

```
1 from subprocess import run
2 run(["fswebcam", "-c", "/home/ahg/bild.cfg"])
```

Die Methode `run` des Moduls `subprocess` wird in Abschnitt 8.2.5 (Seite 335) erläutert.

Wenn `ffmpeg` installiert ist (siehe Seite 78) kann man auch mit

```
ffmpeg -f v4l2 -video_size 1280x480 -i /dev/video0 -frames 1 bild.jpg
```

ein Bild aufnehmen. Mit dem Befehl `man ffmpeg` erfährt man mehr über `ffmpeg`.

6.1.3 PDF-Dokumente und E-Books

PDF-Dokumente anzeigen mit fbgs

Zur graphischen Darstellung einer PDF-Datei kann man das Programm `fbgs` benutzen, das nach Installation des Pakets `fbi` verfügbar ist. `fbgs` erzeugt mithilfe von Ghostscript eine Bilddatei und zeigt diese dann mit dem Programm `fbi`. Eine PDF-Datei `/home/ahg/latex/was.pdf`, die mit dem Passwort `pw` geschützt ist, wird mit dem Befehl

```
fbgs -p pw /home/ahg/latex/was.pdf
```

angezeigt. Ist die PDF-Datei nicht durch ein Passwort geschützt, entfällt die Option `-p pw`. Statt der gesamten PDF-Datei kann ein Seitenbereich gezeigt werden; `-fp` i bestimmt die erste Seite i (wobei i eine Seitenzahl ist) und `-lp` j bestimmt die letzte Seite j.

Die Standard-Auflösung beträgt nur 75 dpi. Mit der Option `-l` wird die Auflösung auf 100 dpi erhöht, mit `-xl` auf 120 dpi und mit `-xxl` auf 150 dpi. Meistens wird man die Option `-xxl` brauchen oder man nimmt die Option `-r` n und setzt die Auflösung auf n dpi, zum Beispiel mit n = 300. Höhere Auflösung vergrößert das Bild.

Wichtige Befehle, die durch Drücken einer Taste gegeben werden, sind

h Hilfe zu den Tastenbefehlen ein/aus
j eine Seite vor
k eine Seite zurück
v Anzeige einer Statuszeile ein/aus
\+ Vergrößerung der Seite
– Verkleinerung der Seite
↑ ↓ Verschiebung einer Seite, die nicht auf den Bildschirm passt
q (quit) Programm `fbgs` beenden

PDF-Dokumente bearbeiten mit poppler-utils

Das Paket `poppler-utils` wird mit dem Befehl

```
sudo apt install popper-utils
```

installiert. Es enthält Programme, die Information über PDF-Dokumente ausgeben oder PDF-Dokumente (oder Teile davon) in ein anderes Format wandeln.

Information zu einer PDF-Datei `datei.pdf` erhält man mit dem Befehll

```
pdfinfo datei.pdf
```

Der Textinhalt einer unverschlüsselten PDF-Datei `datei.pdf` wird mit dem Befehl

```
pdftotext datei.pdf textdatei.txt
```

in eine Textdatei `textdatei.txt` geschrieben, welche allerdings nachbearbeitet werden muss. Einzelne Seiten einer unverschlüsselten PDF-Datei `datei.pdf` erhält man mit

```
pdfseparate datei.pdf einzel-%d.pdf
```

Jede Seite der mehrseitigen PDF-Datei `datei.pdf` wird in einer Datei einzel-%d.pdf gespeichert, wobei %d automatisch durch die Seitenzahl ersetzt wird. Der Name „einzel“ der hergestellten Dateien ist beliebig, aber das Namensmuster „einzel-%d.pdf“ der hergestellten Dateien muss „%d“ enthalten.

Braucht man nur einen Seitenbereich, zum Beispiel Seiten 2 bis 5, gibt man den Befehl

```
pdfseparate -f 2 -l 5 datei.pdf einzel-%d.pdf
```

ein. Anfangsseite und Endseite können gleich sein, so dass

```
pdfseparate -f 3 -l 3 datei.pdf einzel-%d.pdf
```

nur eine Datei für Seite 3 erzeugt. Allerdings kann die so hergestellte PDF-Datei fast so groß sein wie die ursprüngliche PDF-Datei, weil eingebettete Daten zur Beschreibung des Dokuments (zum Beispiel Schriftarten) auf jede hergestellte PDF-Datei übertragen werden. Man kann eine Verkleinerung der hergestellten PDF-Dateien erreichen, wenn man die ursprüngliche PDF-Datei vorher mit dem Programm Ghostscript optimiert (was wir hier nicht behandeln).

Unverschlüsselte PDF-Dateien, zum Beispiel `dat1.pdf` und `dat2.pdf`, können mit

```
pdfunite dat1.pdf dat2.pdf vereinigt.pdf
```

zu einer PDF-Datei `vereinigt.pdf` zusammengefasst werden. Der Name der hergestellten PDF-Datei ist beliebig, muss aber angegeben werden,da ansonsten die letztgenannte Datei überschrieben wird.

Ein sogenanntes PDF-Portfolio ist das besondere, nicht allgemein spezifizierte Format eines Softwareunternehmens („Lehmziegel“). Die darin enthaltenen Dokumente können mit dem Programm `pdfdetach` entpackt werden. Zunächst richtet man ein Verzeichnis (zum Beispiel `pfverz`) für die PDF-Portfoliodatei (zum Beispiel `pf.pdf`) ein

```
mkdir ~/pfverz
```

und dann speichert man den entpackten Inhalt in diesem Verzeichnis.

```
pdfdetach -saveall -o ~/pfverz pf.pdf
```

Die entpackten PDF-Dateien können dann mit `fbgs` angesehen werden (siehe oben).

Ein einfaches PDF-Portfolio `pf.pdf` kann auch mit dem Ghostscript-Befehl

```
gs -dBATCH -dNOPAUSE -sDEVICE=png16m -r300 -sOutputFile=pf.png pf.pdf
```

in ein Bild im PNG-Format umgewandelt werden. Wenn man zusätzlich die Option `-dSAFER` verwendet, wird Ghostscript im Sicherheitsmodus aufgerufen. Dies verhindert das unbeabsichtigte Überschreiben oder Löschen bereits vorhandener Dateien.

PDF-Dokumente bearbeiten mit PDFtk

Ein weiteres Paket zur Bearbeitung von PDF-Dateien ist `pdftk`. Der Befehl

```
pdftk f1.pdf f2.pdf cat output f_merged.pdf
```

fügt zum Beispiel zwei Dateien `f1.pdf` und `f2.pdf` zu einer `f_merged.pdf` zusammen. Folgender Befehl dreht alle Seiten einer PDF-Datei `f1.pdf` um 180° und speichert das Ergebnis in einer PDF-Datei `f2.pdf` (die nicht gleich `f1.pdf` sein darf):

```
pdftk f1.pdf cat 1-enddown output f2.pdf
```

Die Drehung erfolgt relativ zur aktuellen (sichtbaren) Ausrichtung des Dokuments. (In den Metadaten des PDF-Dokuments ist auch die unsichtbare ursprüngliche, das heißt: bei der Erzeugung vorliegende, Ausrichtung gespeichert.)

eBooks

Ein eBook ist ein digitalisiertes Buch. Dateien im Format PDF können mit dem Programm `fbgs` gelesen werden, siehe oben, Seite 179. Für die besonderen eBook Formate Epub (Dateiendungen `.epub`, `.epub3`), FictionBook (`.fb2`), Mobipocket (`.mobi`) und AZW3 (`.azw`, `.azw3`) kann man in der Linux Konsole das eBook-Leseprogramm `epy` von Benawi Adha benutzen. Es wird mit

```
pip3 install --user epy-reader --break-system-packages
```

installiert und mit dem Befehl

```
epy ~/docum/buch.epub
```

wird das eBook geladen, dessen Dateipfad angegeben wurde. Man kann sich mit Pfeiltasten und der Leertaste im Text bewegen und mit der Taste `q` verlässt man das Programm. Das Programm merkt sich die Position im Text und setzt beim nochmaligen Aufruf an der alten Stelle fort. Die Konfigurationsdatei `/home/ahg/.epy/configuration.json` im Datenformat JSON ermöglicht Anpassungen.

Im Projekt Gutenberg-DE werden deutschsprachige Bücher im Format HTML (Dateiendung `.html`) kostenlos angeboten. Sie können mit einem Browser wie w3m (siehe Seite 209) gelesen werden. Mit dem Befehl

```
w3m https://www.projekt-gutenberg.org/info/texte/lesetips.html
```

öffnet man eine Webseite mit Buchempfehlungen des Projekts Gutenberg-DE.

6.1.4 Bildbearbeitung mit ImageMagick

Das Paket `imagemagick` enthält zahlreiche Befehle zur Erstellung und Bearbeitung von Bildern. Bilder sind in besonderen Formaten gespeichert, unter anderem: jpeg (jpg), ppm, gif, tiff, xwd, bmp und png. Dies erkennt man meistens an der Endung der Datei, zum Beispiel `.jpg` oder `.png`. Das Bildformat wird (wenn nicht durch eine Option vom Nutzer angegeben) automatisch bestimmt, bei Ausgabedateien anhand der Datei-Endung. Rasterbilder sind aus quadratischen Bildpunkten aufgebaut, die Pixel genannt werden.

Die Bildgröße wird in Pixeln als Breite mal Höhe angegeben, wobei der Buchstabe `x` für „mal“ steht.

Information zu einer Bilddatei `b1.png` liefert der Befehl

```
identify b1.png
```

oder, mit ausführlichen Angaben zur Bilddatei,

```
identify -verbose b1.png
```

Bei der Erstellung einer Bilddatei können beschreibende Angaben mit Schlüsselworten in der Datei gespeichert werden. Welche Information möglich ist, hängt vom Bildformat ab. Für PNG sind das unter anderem Angaben zu `Author` und `Title`. Zum Beispiel gibt

```
identify -verbose plot.png | grep "Author"
```

Information zum Urheber eines PNG-Bilds `plot.png`, falls diese Angabe gemacht wurde.

Ein mächtiger Befehl zur Bildbearbeitung ist `convert`. Mit

```
convert b1.png b2.jpg
```

wird eine Bilddatei mit geändertem Format erzeugt.

Eine Spiegelung des Bilds (hier in Datei `b1.png`), die oben und unten vertauscht, und Speicherung in Datei `b2.png` erreicht man durch

```
convert -flip b1.png b2.jpg
```

Entsprechend vertauscht die Option `-flop` links und rechts. Beide Optionen zusammen (`-flip -flop`) bewirken eine halbe Drehung (also um 180°).

Bei Photos (hier in Datei `b1.png`, Ausgabe in Datei `b2.png`) ist es meistens gut, mit

```
convert -normalize b1.png b2.jpg
```

die Hell-Dunkel-Verteilung so zu ändern, dass die hellsten Stellen weiß und die dunkelsten Stellen schwarz werden.

Will man die Größe eines Rasterbilds ändern, zum Beispiel von 720x720 (Bilddatei `b1.png`) auf 200x100 (Bilddatei `b2.png`), gibt man den Befehl

```
convert b1.png -resize 200x100\! b2.png
```

ein. Die Zeichen `\!` bewirken, dass das Seitenverhältnis geändert wird. Ohne sie würde ein Bildformat von 100x100 erzeugt, um das Seitenverhältnis zu bewahren. Eine prozentuale Größenänderung (hier auf halbe Seitenlängen) ergibt der Befehl

```
convert b1.png -resize 50% b2.png
```

wobei die Seitenlängen allerdings immer auf ganzzahlige Werte gerundet werden.

Leicht unscharfe Bilder, oder Bilder deren Größe geändert wurde, kann man oft durch „Schärfung“ verbessern. Zum Beispiel (Eingabedatei `b1.png`, Ausgabedatei `b2.png`):

```
convert -sharpen 1 b1.png b2.png
```

wobei für den „Schärfungsgrad“ (hier 1) auch ein höherer Wert möglich ist.

Eine Invertierung der Farben, also die Ersetzung der Farbe jedes Bildpunkts durch die Komplementärfarbe gelingt mit dem Befehl

```
convert b1.png -channel RGB -negate b2.png
```

Damit wird zum Beispiel Weiß zu Schwarz, Hellgrau zu Dunkelgrau und umgekehrt. Will man ein Bild erzeugen, das nur Schwarz oder Weiß enthält, kann man `convert` mit der Option `threshold` verwenden und eine Schwelle in Prozent angeben. Beispiel:

```
convert b1.png -threshold 75% b2.png
```

Alle Bildpunkte, deren „Helligkeit“ (die wir hier nicht definieren) oberhalb der Schwelle liegt, werden Weiß, die anderen Schwarz. Je höher die Schwelle ist, desto größer wird der Schwarzanteil. Den besten Schwellenwert erhält man durch probieren. Es ist nicht notwendig, aber besser, wenn man Farbbilder vorher in Graustufenbilder verwandelt.

Ein einfarbiger Rahmen der Breite 5 pixel um ein Bild (in Datei `b1.png`) wird mit

```
convert b1.png -bordercolor SeaGreen -border 5x5 b2.png
```

erzeugt (und das Ergebnis in der Datei `b2.png` gespeichert). Die Option `-bordercolor SeaGreen` bestimmt die Farbe SeaGreen (HEX Code: #2E8B57) als Rahmenfarbe. Lässt man diese Option weg, wird die Standardrahmenfarbe, ein sehr helles Grau (HEX Code: #DFDFDF) genommen. Um diesen Rahmen wegzurasieren, gibt man den Befehl

```
convert b2.png -bordercolor -shave 5x5 b3.png
```

Um zwei Bilder (hier in den Dateien `b1.png` und `b2.png`) waagerecht nebeneinander zu setzen und in einer Bilddatei (hier `b3.png`) zu speichern, gibt man die Befehlzeile

```
convert +append b1.png b2.png b3.png
```

ein. Entsprechend setzt man mit

```
convert -append b1.png b2.png b3.png
```

zwei Bilder senkrecht untereinander.

Der Befehl `composite` erlaubt es, ein kleines Bild (zum Beispiel ein Logo, hier in Datei `b1.png`) in ein größeres Bild (hier in Datei `b2.png`) zu setzen und das Ergebnis in einer neuen Datei (hier `b3.png`) zu speichern:

```
composite -gravity SouthEast b1.png b2.png b3.png
```

Die Option `-gravity SouthEast` bestimmt die Lage des kleinen Bildes, hier in der unteren rechten Ecke. Entsprechend würde `-gravity North` das kleine Bild nach oben und `-gravity Center` in die Mitte des größeren Bilds setzen.

6.1.5 Mediaplayer mpv und MPlayer

mpv

Der Mediaplayer mpv ist ist für die Wiedergabe von Audio und Video, auch mit „Hardwarebeschleunigung“ gut geeignet und für die Linux Konsole einfacher und zuverlässiger einzurichten als VLC. Man installiert ihn mit dem Befehl

```
sudo apt install mpv
```

und kann damit Audio- und Videodateien, Internetradio und Internetfernsehen und Youtube-Videos gut abspielen. Nur bei Audio-CDs, VCDs und DVDs ist VLC vielleicht besser geeignet. Der Radiosender `UK 1940s` wird mit dem Befehl

```
mpv https://1940sradio1.co.uk:8100/1
```

gespielt und ein Youtube-Video (Peggy March: „I Will Follow Him“, 1963) wird mit

```
mpv https://www.youtube.com/watch?v=X9Qy9EXaOo0
```

abgespielt.

Optionen mit denen man Information zum erhält sind (unter anderem):

--h zeigt eine kurze Liste einiger Optionen
--h=*string* zeigt eine Liste von Optionen, welche die Zeichenkette *string* enthalten
--list-options zeigt eine lange Liste aller verfügbaren Optionen
--ao=help zeigt eine Tabelle der Liste der Teiber für die Audio-Ausgabe
--vo=help zeigt eine Tabelle der Liste der Teiber für die Video-Ausgabe

Wichtige Optionen für die Ausgabe von Audio und Video mit `mpv` sind

--no-config lade nicht Konfigurationsdateien, Kommandozeilenoptionen wirken aber
--ao=*treiber* verwende *treiber* für die Audio-Ausgabe (`alsa`, `pipewire`, `pulse`)
--audio-device=*treiber* Audiotreiber *treiber* mit Optionen (siehe Seite 73)
--playlist=*datei* verwende die playlist in der Datei *datei*
--loop beim Abspiel einer Datei: unendliche Schleife
--loop-playlist=inf beim Abspiel einer playlist: unendliche Schleife
--start=*hh:mm:ss* Wiedergabe ab der Position (Stunden, Minuten, Sekunden)
--sub-file=*datei* verwende die Untertitel in der Datei *datei*
--vo=*treiber* verwende *treiber* für die Video-Ausgabe (`caca`, `drm`, `gpu`, `sdl`)
--video-unscaled=yes unterlasse die bildschirmfüllende Vergrößerung des Bilds
--video-zoom=*zahl* Bildvergößerung (zoom), abhängig von *zahl* (siehe manpage)
--video-pan-y=*zahl* vertikale Verschiebung, abhängig von *zahl* (siehe manpage)
--video-pan-x=*zahl* horizontale Verschiebung, abhängig von *zahl* (siehe manpage)
--volume=*zahl* Lautstärke (volume), meistens $0 \leq zahl \leq 100$, Standard: 100
--volume-max=*zahl* Maximale Lautstärke (volume), Standard: 130
--screenshot-format=png PNG für Bildschirmphotos statt JPEG (Standard)

--ytdl-raw-options=*key/value list* Schlüssel-Wert-Liste für yt-dlp Optionen
--cdrom-device=*path* spiele Audio-CD über Gerät mit dem Pfad *path* (`/dev/cdrom`)
--dvd-device=*path* spiele eine DVD über ein Gerät mit dem Pfad *path* (`/dev/dvd`)
--quiet weniger Textausgabe auf dem Bildschirm, keine Statuszeile
--really-quiet noch weniger Textausgabe auf dem Bildschirm als mit `--quiet`

Wenn die Audio- und Videotreiber nicht als Option vorgegeben werden, verwendet `mpv` meistens `gpu` als Videotreiber (GPU-accelerated video output) und entweder `alsa` (für den soundserver ALSA) oder `pipewire` (für den soundserver Pipewire) oder `pulse` (für den soundserver PulseAudio) als Audiotreiber. Die automatische Treiberwahl ist meistens richtig. Für besondere Einstellungen des ALSA sound servers siehe Seite 73.

Mit `gpu` wird die „Hardwarebeschleunigung“ der GPU des Raspberry Pi für das Abspiel von Videos genutzt; `sdl` als Videotreiber erlaubt zum Beispiel den Wechsel zum Vollbild mit der `f` Taste (siehe unten, Steuerung über die Tastatur). Bei manchen Optionen nimmt `mpv` automatisch `sdl` als Videotreiber und man muss die Option `--vo=gpu` verwenden, wenn man den Videotreiber `gpu` haben will.

Wenn das Paket `libcaca0` installiert ist, kann man auf schnellen Raspberry-Modellen `caca` als „Videotreiber“ nehmen, um eine farbige Darstellung mit ASCII-Art zu erzeugen. Um mit `caca` eine annehmbare Auflösung zu erreichen, sollte man aber den Font der Schriftzeichen auf kleine quadratische Zeichen verkleinern (siehe Seite 190). Eine Alternative zu ist `caca` ist der „Videotreiber“ `tct`, der Quadrate als Ausgabe-Zeichen verwendet. Auch hierfür sollte man den Font verkleinern.

Durch eine Datei `~/.mplayer/config` kann die Konfiguration von `mpv` für den aktuellen Nutzer bewirkt werden (wenn sie angelegt wurde). In jeder Zeile kann eine Option nach dem Schema --Schlüssel=Wert geschrieben werden. Das Zeichen # leitet einen Kommentar ein, der mit dem Zeilenende abgechlosen wird. Die Einstellungen der Konfigurationsdatei können durch Optionen beim Aufruf von `mpv` überschrieben werden.

Die Steuerung von `mpv` kann über die Tastatur erfolgen:

9 Lautstärke verringern
0 Lautstärke vergrößern
< 1 Stück zurück, wenn eine playlist abgespielt wird
> oder ENTER 1 Stück vorwärts, wenn eine playlist abgespielt wird
m Stummschaltung (mute), nochmals: Aufhebung der Stummschaltung
O aktuelle Position und Gesamtdauer eines Videos ein- oder ausblenden
p Pause (auch mit der Leertaste), nochmals: weiter
. Pause oder, wenn Pause ist, ein Videobild weiterrücken und pausieren
, Pause oder, wenn Pause ist, ein Videobild zurückrücken und pausieren
← 5 Sekunden rückwärts springen
→ 5 Sekunden vorwärts springen
↓ 1 Minute rückwärts springen
↑ 1 Minute vorwärts springen
f Vollbild (fullscreen mode) ein / aus (geht nur bei manchen Videotreibern)
v (quit) Untertitel ein- oder abschalten (wenn diese ausgewählt wurden)

s (quit) screenshot (Bildschirmphoto) machen und als JPG-Datei speichern
S (quit) screenshot (Bildschirmphoto) ohne Untertitel machen und speichern
1 und 2 Bildkontrast (contrast) erniedrigen beziehungsweise erhöhen
3 und 4 Helligkeit (brightness) erniedrigen beziehungsweise erhöhen
5 und 6 Gamma-Wert (gamma) erniedrigen beziehungsweise erhöhen
7 und 8 Farbsättigung (saturation) erniedrigen beziehungsweise erhöhen
L unendliche Schleife des Abspiels (infinite looping) ein / aus
PGUP (Bild↑) zurück zum Kapitelanfang (oder zum vorigen Kapitel)
PGDWN (Bild↓) vorwärts zum Anfang des nächsten Kapitels
q (quit) Abspiel stoppen und mpv beenden
Q (quit) Abspiel stoppen und mpv beenden, zusätzlich: aktuelle Position speichern

Die Konfiguration von `mpv` kann systemweit in einer Datei `/etc/mpv/mpv.conf` erfolgen, deren Einstellungen von der Datei `~/.config/mpv/mpv.conf` des Nutzers überschrieben werden, wenn diese erzeugt wurde. Die Voreinstelllungen der Konfigurationsdateien können durch Optionen beim Aufruf von `mpv` überschrieben werden.

Bei der Wiedergabe einer Datei erscheint im Terminal eine Zeile (terminal status line) die angibt, ob Audio (`A`), Video (`V`) oder beides (`AV`) ausgegeben wird, die aktuelle Spielzeit (Stunden:Minuten:Sekunden), die Gesamtdauer (Stunden:Minuten:Sekunden), der prozentuale Anteil der schon abgespielt wurde, und die ungewollte Zeitverschiebung zwischen Audio und Video, die 0 sein sollte (`AV: 0.000`). Falls die Geschwindigkeit der Wiedergabe verändert wurde, wird auch dies angegeben (nach der Gesamtdauer).

Installation und Konfiguration des MPlayer

Ein bewährter Mediaplayer ist MPlayer, aber es fehlt ihm beim Raspberry Pi die „Hardwarebeschleunigung“, sodass er für Videos schlechter ist als mpv oder VLC (siehe Seite 191). Für die Audiowiedergabe ist er aber gut geeignet.

Bei der Installation des Pakets `mplayer` mit dem Befehl

```
sudo apt install mplayer mplayer-doc
```

werden die erforderlichen Bibliotheken und Hilfsprogramme automatisch mit installiert, sodass der Umfang groß ist. Wenn, wie oben empfohlen, auch `mplayer-doc` installiert wurde und auch ein Browser, zum Beispiel w3m (siehe Seite 209, installiert wurde, kann man die englische Dokumentation im Browser offline (ohne Internetverbindung) lesen:

```
w3m /usr/share/doc/mplayer/HTML/en/index.html
```

Es gibt auch eine deutsche Dokumentation (mit „de“ statt „en“ in obiger URL), aber die ist veraltet. Die neuesten Dokumentationen und Verweise findet man auf der Webseite

http://www.mplayerhq.hu/design7/documentation.html

und auf

http://www.mplayerhq.hu/design7/info.html

werden die wesentlichen Eigenschaften (features) des MPlayers aufgeführt.

Man kann auch Voreinstellungen vornehmen, die nur für Media-Dateien mit einer bestimmten Dateinamenerweiterung, zum Beispiel `.mp3`, oder nur für DVDs gelten. Außerdem können die vorbelegten Tastenfunktionen geändert werden.

Die deutsche Übersetzung der online Dokumentation des MPlayers findet man im HTML-Format auf der Webseite http://www.mplayerhq.hu/DOCS/HTML/de/

Die Konfiguration erfolgt systemweit durch die Datei `/etc/mplayer/mplayer.conf`, deren Einstellungen von der Datei `~/.mplayer/config` des Nutzers überschrieben werden. Diese Voreinstelllungen können durch Optionen beim Aufruf des Mplayers überschrieben werden. Die Datei `/etc/mplayer/mplayer.conf` lassen wir unverändert und schreiben unsere Einstellungen, mithilfe eines Texteditors wie `nano` (siehe Seite 131), in die noch leere Datei `~/.mplayer/config`.

```
# Info and Control
quiet=1
really-quiet=1
nolirc=1
# Audio (output driver: alsa, pulse, sdl, null)
ao="alsa,pulse"
# Video (output driver: fbdev2, aa, sdl, null)
vo="fbdev2"
vf="scale"
framedrop=1
# Buffer
cache=8192
cache-min=2
```

Mit `quiet=1` und `really-quiet=1` wird die Ausgabe von Information und Fehlermeldungen auf dem Bildschirm unterdrückt. Diese Zeilen sollte man also nur dann einfügen, wenn alles richtig funktioniert. Durch `nolirc=1` teilen wir mit, dass wir keine Infrarot-Fernsteuerung benutzen.

Der Audioausgabetreiber muss zum sound server passen, den das Betriebssystem verwendet. Wichtige sind `alsa` (für ALSA) und `pulse` für Pipewire und PulseAudio. Für Pipewire schreiben wir folgende Zeile in die Zeile `ao=pulse,alsa,sdl:aalib` (siehe oben) und für ALSA wäre es entsprechend `ao=alsa,pulse,sdl:aalib`

Der Parameter `cache` gibt die Größe des Zwischenspeichers in KiB an und der Wert von `cache-min` bestimmt, wie viel Prozent des Zwischenspeichers (höchstens 99) gefüllt sein müssen, bevor die Wiedergabe beginnt. Für langsame Medien, zum Beispiel CDs, sollte man große Werte wählen, zum Beispiel `cache=16384` und `cache-min=90`, aber natürlich muss es in den Arbeitsspeicher passen. Für Internetradiosender sollten aber niedrigere Werte gewählt werden, zum Beispiel `cache 8192` und `cache-min=2`, da sich der Beginn der Wiedergabe verzögert.

Am Ende der Konfigurationsdatei sollte ein Zeilenumbruch erfolgen. Bei der Wiedergabe von CDs sollte man die oben genannten höheren Werte von `cache` und `cache-min` als Optionen beim Aufruf des MPlayers übergeben.

Optionen und Tastensteuerung des MPlayer

Wichtige Optionen für die Ausgabe von Audio und Video mit dem MPlayer sind

-ao alsa Audioausgabe über ALSA
-ao pulse Audioausgabe über Pipewire oder PulseAudio
-vo fbdev2 Videoausgabe über den Framebuffer
-cache x Zwischenspeicher mit x kb (Standard ist `nocache`)
-cache-min x Start erst nachdem der Zwischenspeicher zu x % gefüllt ist
-dumpaudio x Speicherung eines Audiostroms (Datei mit `-dumpfile` x angeben)
-dumpfile x Pfad zu einer Datei für die Speicherung eines Datenstroms
-dumpstream x Speicherung eines Datenstroms (Datei mit `-dumpfile` x angeben)
-nolirc Abschalten der Fernbedienungsmöglichkeit (Linux Infrared Remote Control)
-quiet weniger Hinweise, Unterdückung der Statuszeile bei Videoausgabe
-really-quiet deutlich weniger Hinweise auf dem Bildschirm (Audio und Video)
-v mehr Hinweise auf dem Bildschirm (aktuelles Lied bei einem Radiosender)

Die Steuerung des MPlayers kann über die Tastatur erfolgen:

9 Lautstärke verringern
0 Lautstärke vergrößern
(bei Audiowiedergabe mit Stereophonie: Balance zum linken Kanal schieben
) bei Audiowiedergabe mit Stereophonie: Balance zum rechten Kanal schieben
m Stummschaltung (mute), nochmals: Aufhebung der Stummschaltung
p Pause (auch mit der Leertaste), nochmals: weiter
← 10 Sekunden rückwärts springen (bei Audiodateien)
→ 10 Sekunden vorwärts springen (bei Audiodateien)
↓ 1 Minute rückwärts springen (bei Audiodateien)
↑ 1 Minute vorwärts springen (bei Audiodateien)
< 1 Stück zurück, wenn eine playlist abgespielt wird
> 1 Stück vorwärts, wenn eine playlist abgespielt wird
q (quit) Abspiel stoppen und Mplayer beenden

Audio mit MPlayer

Wir gehen davon aus, dass die Konfigurationsdatei `~/.mplayer/config` bereits erstellt wurde. Zum Abspielen einer Audiodatei `~/audio/lied.mp3` genügt dann der Befehl

```
mplayer ~/audio/lied.mp3
```

Unabhängig von der Konfigurationsdatei bewirkt die Option `-ao alsa`, dass die Audioausgabe zum ALSA-Treiber gelenkt wird. Das Ausgabegerät, zum Beispiel eine USB-Soundkarte, könnte zusätzlich angegeben werden (siehe Seite 73).

Das Abspielen einer playlist (siehe oben auf Seite 197) erfordert die Option `-playlist`.

```
mplayer -playlist ~/audio/lieder.m3u
```

Um die Dateien einer playlist in zufälliger Reihenfolge zu spielen, gibt man den Befehl

```
mplayer -shuffle -playlist ~/audio/lieder.m3u
```

und wenn es zusätzlich in einer ewigen Schleife spielen soll, gibt man den Befehl

```
mplayer -loop 0 -shuffle $(cat ~/audio/lieder.m3u)
```

Für das Abspielen von Internetradiosendern braucht man die Internetadresse (URL) des Audiodatenstroms und muss das Datenformat kennen. Bietet der Sender, wie in folgendem Beispiel (RBB Inforadio), einen mp3-Stream, genügt der Befehl

```
mplayer http://inforadio.de/livemp3
```

Verwendet der Sender das Format einer playlist (Wiedergabeliste), dann braucht man die Option `-playlist`, wie in folgendem Beispiel mit Radio Caroline (48 kBit/s). Da der Befehl lang ist, schreiben wir ihn in die Textdatei `rc`.

```
mplayer -ao alsa -cache 128 -playlist \
http://sc2.radiocaroline.net:8010/listen.pls
```

Dann machen wir die Datei `rc` ausführbar.

```
sudo chmod a+x rc
```

und danach können wir den Radiosender mit dem Befehl

```
./rc
```

starten. Wenn man nicht weiß, ob ein Sender das Format einer playlist verwendet, muss man beide Möglichkeiten (ohne und mit der Option `-playlist`) ausprobieren.

Man kann die Tonspur eines Radiosenders aufzeichnen und in einer Datei speichern, bis die Aufzeichnung mit der Taste `q` beendet wird:

```
mplayer RADIOSENDER -dumpfile DATEINAME -dumpaudio
```

wobei für RADIOSENDER die URL des Radiosenders und für DATEINAME der (beliebige) Name der Audiodatei (mit der Tonspur) eingetragen werden. Die Audiodatei kann später mit dem MPlayer oder VLC abgespielt werden. Für Sender im Format einer playlist braucht man die Option `-playlist`.

Hat man einen CD-Spieler über USB angeschlossen (am besten mit einem aktiven USB-Hub), kann man eine Audio-CD mit dem Befehl

```
mplayer cdda:// -cache 8192 -cache-min 90
```

abspielen. Es gibt auch eine Option `-cdrom-device`, mit der das CD-Gerät angegeben werden kann, aber das ist meistens nicht nötig (der Standardwert ist `/dev/cdrom`). Beim Abspielen zeigt die letzte Prozentangabe in der Statuszeile an, wie viel des Zwischenspeichers gefüllt ist; meistens sind es etwa 49%.

Video mit MPlayer

Der Mediaplayer `mplayer` unterstützt viele Video-Codecs und er gibt auch animierte GIFs wieder. Da dem `mplayer` beim Raspberry Pi die „Hardwarebeschleunigung“ fehlt, sollte die Wiedergabe, soweit möglich, mit der geringsten Bildauflösung (worst) erfolgen. Das Abspiel von Videodateien ist so mit dem `mplayer` möglich, aber für Internetfernsehen benötigt man mindestens einen Raspberry Pi 4 (und auch dann ist es schlecht).

Wir gehen davon aus, dass die Konfigurationsdatei `~/.mplayer/config` bereits erstellt wurde. Zum Abspielen einer Videodatei `~/video/movie.mp4` genügt dann

```
mplayer ~/video/movie.mp4
```

Das Format der Wiedergabe wird allerdings nicht an die Bildschirmgröße angepasst. Um ein Vollbild zu erhalten, muss man die Bildschirmgröße kennen. Der Befehl

```
fbset -s | grep mode | awk '{print $2}'
```

liefert die Information. Für eine Größe von 1920x1080 zeigt

```
mplayer -quiet -zoom -x 1920 -y 1080 ~/video/movie.mp4
```

den Film in voller Bildschirmgröße, vorausgesetzt der Raspberry Pi schafft es. Man sollte einen Raspberry Pi 4 haben, mindestens einen Raspberry Pi 3. Falls eine schwarze Stauszeile am unterern Bildrand stört, kann man die Option `-quiet` hinzufügen.

mplayer und ASCII-Art

Bilder lassen sich auch ohne Graphikunterstützung darstellen, indem statt einzelner Bildpunkte (Pixel) Schriftzeichen zeilenweise ausgegeben werden. Das ist dann Kunst und heißt ASCII-Art. Die Glyphen (Zeichenformen) des Fonts der Konsole erzeugen das Bild. Mit `mplayer` und der Bibliothek AAlib können so ohne den framebuffer Videos dargestellt werden. Es ergibt eine witzige, doch viel zu ungenaue Wiedergabe. Man sollte nicht mehr als einen interessanten Effekt erwarten.

Die Umwandlung der Bildpunkte in Zeichen ist sehr rechenintensiv. Damit Bild und Ton synchron wiedergegeben werden, ist es daher oft notwendig, die Anzahl der Zeilen und Spalten des sichtbaren Terminals zu reduzieren. Mit leistungsfähigeren Modellen des Raspberry Pi muss man das Terminal weniger stark verkleinern. Die aktuelle Zeilenzahl des Terminals erfahren wir mit

```
tput lines
```

und die aktuelle Spaltenzahl mit

```
tput cols
```

Für ein Video mit 640 × 360 Bildpunkten (diese Information erhält man durch den Befehl `mediainfo`, siehe Seite 197) nimmt man zum Beispiel 64 Spalten und 36 Zeilen. Falls das Terminal weniger als 36 Zeilen hat oder der Rechner sehr schwach ist, nimmt man zum Beispiel 32 Spalten und 18 Zeilen. Nun ändern wir den Font der Schriftzeichen (siehe Abschnitt 2.1.2 auf Seite 47) auf kleine quadratische Zeichen. Nach dem Befehl

```
sudo dpkg-reconfigure console-setup
```

nimmt man im Menü folgende Einstellungen vor:

```
UTF-8
Vermutlich optimaler Zeichensatz / Guess optimal character set
VGA
8 x 8
```

Anschließend geben wir, um das Terminal zu verkleinern, in unserem Beispiel den Befehl

```
stty cols 64 rows 36
```

Bei leistungsfähigeren Modellen des Raspberry Pi kann man diese Veränderung der Zeilenzahl und Spaltenzahl einfach weglassen und erhält ein größeres Bild.

Jetzt kann man eine Videodatei, zum Beispiel `video.mp4` im Arbeitsverzeichnis, mit

```
mplayer -vo aa -quiet -framedrop video.mp4
```

abspielen. Die Darstellung ist nur schwarz-weiß. Die Option `-quiet` unterdrückt die Ausgabe der Statuszeile. Mit der Option `-framedrop` erlauben wir weniger leistungsfähigen Rechnern, dass Bilder des Videos ausgelassen werden.

Zeilen und Spalten des Terminals werden beim Neustart des Rechners zurückgesetzt. Man kann sie auch mit dem Befehl `stty` auf den ursprünglichen Wert setzen (siehe oben). Um danach wieder mit der gewohnten Schriftart im Terminal zu arbeiten, stellt man anschließend wieder einen größeren Font ein, wie in Abschnitt 2.1.2 beschrieben.

Im Prinzip geht es auch bunt. Ersetzt man die Option `-vo aa` durch `-vo caca`, erhält man eine farbige Darstellung. Allerdings ist das Ergebnis nicht wirklich überzeugend.

6.1.6 Mediaplayer VLC

Die Installation des Mediaplayers VLC wurde bereits in Abschnitt 3.1.1 auf Seite 78 beschrieben. VLC ist vielseitig, umfangreich und gut für verschiedene Betriebssysteme mit graphischer Oberfläche geeignet, kann aber auch von der Kommandozeile gesteuert werden. Die Konfiguration von VLC für die Kommandozeile ist leider schwierig und die umfangreiche Dokumentation des VLC ist hier weniger hilfreich.

Einstellungen können in einer Konfigurationsdatei erfolgen oder durch Optionen, die beim Aufruf übergeben werden. Empfehlenswert ist die Erstellung einer ausführbaren Datei (Shellscript oder Pythonprogramm) für jede Anwendungsart. Mithilfe eines Texteditors können die Optionen leicht gesetzt und verändert werden und man kann auf die Bearbeitung der umfangreichen Konfigurationsdatei verzichten. Auf diese Weise nutzt man den Vorteil der Kommandozeile, Anwendungen mit kurzen, starken, leicht anpassbaren Befehlen ausführen zu können.

Für die Steuerung des VLC über ein Python-Programm gibt es ein Modul, das in Abschnitt 8.2.4 auf Seite 327 beschrieben wird. Hier stellen wir zunächst die grundsätzlichen Steuerungsmöglichkeiten über die Schnittstellenmodule der Kommandozeile und

die Optionen vor, dann die Verwendung von Shellscripts. Schließlich folgen Hinweise zur Konfigurationsdatei.

Die Steuerung des VLC erfolgt über ein Schnittstellenmodul (interface module). Im Terminal gibt es `rc`, `ncurses`, `telnet` und `dummy`. Die ersten drei ermöglichen eine Kommunikation des Nutzers mit VLC. Das `dummy interface` läuft dagegen nach dem Aufruf in der Regel ohne Kommunikation. `dummy`, `rc` und `ncurses` behandeln wir unten getrennt, `telnet` betrachten wir nicht.

Die Audio-Wiedergabe läuft gut mit allen genannten Schnittstellenmodulen und dem Python-Modul `python-vlc`. Mit `ncurses` kann VLC auch Video-Dateien abspielen. Mit den anderen Schnittstellenmodulen und dem Python-Modul läuft die Wiedergabe von Videos dagegen unter dem aktuellen Betriebssystem unzuverlässig. Die Gründe sind mir nicht bekannt. Oft gibt es nur eine Audio-Ausgabe und eine Fehlermeldung. Nach einem solchen Fehler kann es notwendig sein, das Betriebssystem neu zu starten, bevor man das Schnittstellenmodul `ncurses` verwenden kann.

Man kann VLC eine playlist-Datei im Format M3U (siehe Seite 197) übergeben. Das geht mit allen Schnittstellenmodulen, zum Beispiel mit `ncurses` (siehe Seite 193).

```
vlc --intf ncurses /home/ahg/audio/mymusic.m3u
```

dummy interface

Das `dummy` interface kann Audio und grundsätzlich auch Video abspielen, stellt aber während des Abspiels keine Steuerungsmöglichkeit bereit. Für das Abspiel von Video-Dateien sollte man besser das `ncurses`-Modul benutzen, da die Wiedergabe von Videos mit dem `dummy` interface unter dem aktuellen Betriebssystem oft fehlerhaft ist.

Eine Audiodatei, zum Beispiel `/home/ahg/audio/lied.mp3`, wird mit dem Befehl

```
vlc --intf dummy --play-and-exit /home/ahg/audio/lied.mp3
```

abgespielt, wobei die Ausgabe über einen Audioausgang erfolgt, den VLC als Standard-ausgang ansieht (der analoge Audioausgang oder HDMI). Statt `vlc -intf dummy` kann man `vlc -I dummy` oder `cvlc` schreiben.

Die Option `-play-and-exit` sorgt dafür, dass VLC nach dem Abspiel automatisch beendet wird. Ansonsten müsste man VLC mit der Tastenkombination `Ctrl-C` verlassen, um zur Kommandozeile zurückzukehren (siehe Seite 12).

Ein Video wird mit VLC wie ein Audio abgespielt (abgesehen von Fehlern). Gibt es eine Verbindung zum Internet, kann man zum Beispiel einen YouTube-Datenstrom mit dem `dummmy interface` wiedergeben (wenn das Raspberry Pi Modell es schafft).

```
cvlc --alsa-audio-device=hw:1,0 --gain 0.1 http://youtu.be/V0I5eglJMRI
```

Mit der Option `gain=0.1` wird es leiser (andere Werte zwischen 0 und 8 sind auch möglich). `-alsa-audio-device=hw:1,0` legt das Audiogerät für die Ausgabe mit ALSA fest (siehe Seite 71). Weitere Optionen besprechen wir später (auf Seite 194)

rc interface

Ohne graphische Oberfläche ist `rc` das Standard-Interface. Es ermöglicht die Steuerung durch Eingabe von Befehlen über die Tastatur. Beschränkt man sich auf Audioausgabe, sind die Eingaben im Terminal sichtbar. Bei Videoausgabe mit Vollbild werden die Eingaben vom Video verdeckt, wirken aber dennoch. Wichtige Befehle sind

get_length Gesamtlänge (Dauer) des derzeitig gespielten Stücks in Sekunden
get_title Nennung des Titels des derzeitig gespielten Stücks
info Information zum stream des derzeitig gespielten Stücks
next Sprung zum nächsten Stück der playlist, wenn eine playlist abgespielt wird
pause Abspiel anhalten beziehungsweise weiterlaufen lassen
playlist Anzeige des Inhalts der playlist, wenn eine playlist abgespielt wird
prev Sprung zum vorigen Stück der playlist, wenn eine playlist abgespielt wird
quit oder kurz `q` Beenden von VLC
repeat fortwährende Wiederholung des derzeitigen Stücks ein- oder ausschalten
status Status des derzeitigen Stücks: Quelle (Datei oder URL) und Lautstärke
volume x Einstellung der VLC-Lautstärke auf den Wert x
voldown Erniedrigung der VLC-Lautstärke um die Standardschrittweite
voldown x Erniedrigung der VLC-Lautstärke um die Schrittweite x
volup Erhöhung der VLC-Lautstärke um die Standardschrittweite
volup x Erhöhung der VLC-Lautstärke um die Schrittweite x

Einen Internetfernsehsender kann man ebenso, zum Beispiel mit dem Befehl

```
vlc https://live.chdrstatic.com/cbn/primary/1.m3u8
```

abspielen und mit `q` und `ENTER` beenden. Für das Abspiel von Video-Dateien sollte man besser das `ncurses`-Modul benutzen, da die Wiedergabe von Videos mit dem `rc` interface unter dem aktuellen Betriebssystem oft fehlerhaft ist. Mit Audio-Dateien gibt es keine Probleme.

ncurses interface

Man kann eine Audiodatei, zum Beispiel `/home/ahg/audio/lied.mp3`, mit dem Befehl

```
vlc --intf ncurses /home/ahg/audio/lied.mp3
```

abspielen, wobei die Ausgabe über einen Audioausgang erfolgt, den VLC als Standardausgang ansieht, vermutlich der analoge Audioausgang.

Die Option `-intf ncurses` (Kurzform `-I ncurses`) wählt das `ncurses` interface. Es ermöglicht die Steuerung über Tasten. Am wichtigsten ist `h`, weil damit die Hilfe angezeigt wird, welche die anderen Tastenfunktionen erklärt. Einige wichtige Tastenbefehle für Audio und Video sind

h Hilfe anzeigen / verbergen
i Information zur Audiodatei anzeigen / verbergen
a Lautstärke vergrößern

z Lautstärke verringern
<space> (Leertaste) Pause, nochmals: weiter
← rückwärts springen (bei Audiodateien)
→ vorwärts springen (bei Audiodateien)
m (mute) während des Abspiels Ton ausschalten oder wieder einschalten
n (next) während des Abspiels zum nächsten Stück der Playlist gehen
p (previous) während des Abspiels zum vorigen Stück der Playlist gehen
P (großes P) Playlist anzeigen / verbergen
↓ in der Anzeige der Playlist nach unten gehen
↑ in der Anzeige der Playlist nach oben gehen
<Enter> Abspiel der Datei, die in der Playlist markiert ist, starten
B (großes B) filebrowser (Suchfeld für Dateien) anzeigen / verbergen
q (quit) Abspiel stoppen und `VLC` beenden

Das Abspiel von Videodateien mit dem `ncurses` Interface des VLC erfolgt ähnlich dem oben beschriebenen Abspiel von Audiodateien. Wir gehen jetzt davon aus, dass die Konfigurationsdatei bereits eingerichtet wurde, siehe Seite 195. Eine Videodatei, zum Beispiel `/home/ahg/video/film.mp4` (im Format MPEG-4) lässt sich dann mit dem Befehl

```
vlc /home/ahg/video/film.mp4
```

abspielen. Um einen Internetfernsehsender zu sehen, zum Beispiel Cheddar News, gibt man dessen URL an.

```
vlc https://live.chdrstatic.com/cbn/primary/1.m3u8
```

Es können aber nur Fernsehsender wiedergegeben werden, die ein passendes Datenformat anbieten und deren Datenrate nicht zu hoch ist (für die Internetverbindung und für den Raspberry Pi). Eine Auswahl findet man im Anhang (Seite 417). Will man einen Sender aus einer playlist mit Internetfernsehsendern auswählen, ist es sinnvoll das automatische Abspiel des ersten Senders der Liste zu verhindern. Dies geht mit der Option `--no-playlist-autostart` (siehe unten, bei VLC Optionen).

VLC Optionen

Beim Aufruf von `vlc` gibt es viele Optionen, die grundsätzlich mit allen Schnittstellenmodulen verwendet werden können. Einige wichtige sind

--version Ausgabe der aktuellen Versionsnummer von VLC
--list Ausgabe einer Liste verfügbarer Module (Kurzform: `-l`) `vlc -l | less`
--advanced --help-verbose --module *m* ausführliche Hilfe zum Modul m
--quiet keine Ausgabe von Information im Terminal
--help Hilfe zu Optionen der Kommandozeile (`vlc --help | less`)
--gain *x* Audioverstärkung (Lautstärke) mit $0 \leq x \leq 8$, Gleitpunktzahl (float)
--playlist-autostart automatischer Start des ersten Eintrags einer playlist (Standard)
--no-playlist-autostart kein automatischer Start des ersten Eintrags einer playlist

--**no-audio** ohne Ton, außer beim streaming (bei der Speicherung in Dateien)
--**aout** ***m*** mit m = audio output module (Wahl des Audioausgabemoduls)
--**no-video** ohne Video-Ausgabe
--**fullscreen** Ausgabe auf ganzem Bildschirm (Kurzform: `-f`)
--**width** ***n*** Breite des Videobilds (Standard: -1, abhängig vom Video)
--**height** ***n*** Höhe des Videobilds (Standard: -1, abhängig vom Video)
--**zoom** ***x*** Vergrößerungsfaktor (Zoom), Gleitpunktzahl (float)
--**random** zufällige Dateiauswahl für das Abspiel
--**loop** ewige Wiederholung der playlist
--**repeat** ewige Wiederholung des aktuellen Stücks
--**cdda-track** ***n*** Track (einzelnes Stück) auf einer Audio-CD oder einer DVD
--**play-and-exit** VLC nach dem Abspiel verlassen (zurück zum Terminal)

Die früher mögliche Option `--volume` n gibt es nicht mehr.

VLC Konfigurationsdatei

Die VLC Konfigurationsdatei für den Nutzer `pi` ist `/home/ahg/.config/vlc/vlcrc` und sie kann mit einem Texteditor wie `nano` bearbeitet werden (siehe Seite 131). Die Konfigurationsdatei ist für Audio und Video zuständig.

Wir nehmen einige Einstellungen für Audio und Video vor. Die Einstellungen in dieser Konfigurationsdatei haben Vorrang vor den allgemeinen Einstellungen des VLC, welche ansonsten für alle Nutzer angewendet werden. Einstellungen, die man über Optionen beim Aufruf des VLC übergibt, haben wiederum Vorrang vor den Einstellungen in der Konfigurationsdatei. Zum Beispiel wird mit der Option `-intf dummy` das sogenannte `dummy interface` verwendet, auch wenn in der Konfigurationsdatei das `ncurses interface` eingestellt wurde. Man sollte aber das `dummy interface` meiden, weil es bei der Videoausgabe zu Fehlfunktionen unter dem aktuellen Betriebssystem kommen kann.

Um die Audio-Ausgabe einzuschalten ersetzen wir die Zeile

```
#audio=1
```

durch die Zeile

```
audio=1
```

Dies ist zwar in der Regel überflüssig, da der Standardwert ohnehin 1 ist, aber wir haben die Möglichkeit, den Wert auf 0 zu setzen und damit die Audioausgabe abzuschalten (warum auch immer). Die Zeile

```
#stereo-mode=0
```

ersetzen wir durch die Zeile

```
stereo-mode=1
```

um die Stereo-Audioausgabe zu aktivieren.

Zur Festlegung des HDMI-Audioausgangs ersetzen wir die Zeile

```
#alsa-audio-device=default
```

durch die Zeile

```
alsa-audio-device=hw:0,0
```

ersetzt. Für den analogen Audioausgang würden wir stattdessen

```
alsa-audio-device=hw:1,0
```

schreiben.

Um (für Videodateien) die Video-Ausgabe einzuschalten ersetzen wir die Zeile

```
#video=1
```

durch die Zeile

```
video=1
```

Dies ist zwar in der Regel überflüssig, da der Standardwert ohnehin 1 ist, aber wir haben die Möglichkeit, den Wert auf 0 zu setzen. In letzterem Fall wird die Videoausgabe abgeschaltet und die Decodierung der Videodaten entfällt. Damit kann man bei schwacher Prozessorleistung zumindest den Ton eines Videos genießen.

Eine Schnittstelle (interface) zwischen VLC und dem Benutzer ermöglicht die Kommunikation mit einem laufenden VLC Programm, zum Beispiel zur Lautstärkeregelung. Sie wird durch ein interface module bereitgestellt. Um das interface module `ncurses` zu verwenden, ersetzen wir die Zeile

```
#intf=
```

durch die Zeile

```
intf=ncurses
```

Nachdem wir das interface module `ncurses` in der Konfigurationsdatei eingestellt haben, brauchen wir die Option `-I ncurses` beim Aufruf des VLC nicht mehr.

Statt einer Audiodatei kann ein Internetradiosender, zum Beispiel Radio 1920, mit der Angabe seiner Internetadresse (URL) und dem Befehl

```
vlc http://stream.laut.fm/radio1920
```

gehört werden.

VLC und Audio-CDs

Ist ein CD-Abspielgerät über USB angeschlossen, kann man eine Audio-CD abspielen. Wir gehen jetzt davon aus, dass die Konfigurationsdatei bereits eingerichtet wurde, siehe Seite 195.

```
vlc cdda://
```

Um nur einen Track, zum Beispiel 3, abzuspielen, gibt man folgenden Befehl:

```
vlc --play-and-exit --cdda-track 3 cdda://
```

6.1.7 Weitere Programme für Audio und Video

mediainfo

Information über eine Audio- oder Videodatei `datei`, zum Beispiel die Dauer (duration) oder, bei Videodateien, die Bildgröße (width und height), erhält man, wenn das Paket `mediainfo` installiert wurde (siehe Seite 79), mit dem Befehl

```
mediainfo datei
```

Audiodaten in das Format MP3 wandeln mit ffmpeg

`ffmpeg` ermöglicht das Umwandeln von Audio- und Videoformaten. Nach der Installation von `ffmpeg` (siehe Seite 78) kann man zum Beispiel mit dem einfachen Befehl

```
ffmpeg -i ~/audio/lied1.wav ~/audio/lied1.mp3
```

aus einer WAVE-Audiodatei eine Audiodatei im Format MP3 erzeugen und mit

```
ffmpeg -i ~/video/lied2.mp4 ~/audio/lied2.mp3
```

kann man die Tonspur einer MP4 Videodatei in einer MP3 Audiodatei speichern. Es gibt natürlich viele Einstellmöglichkeiten über Optionen. Die Dokumentation von `ffmpeg` findet man auf der Webseite https://ffmpeg.org/documentation.html

Erstellung einer playlist-Datei im Format M3U

Eine playlist (Wiedergabeliste) im Format *extended M3U* (erweitertes M3U) kann in einer Textdatei gespeichert sein. Diese man mit einem Editor oder einem besonderen Hilfsprogramm erstellen. Sie könnte zum Beispiel so aussehen:

```
#EXTM3U
#EXTINF:166, ABBA: People Need Love (1972)
/home/ahg/audio/abba_peopleneedlove.mp3
#EXTINF:174, Marianne Rosenberg: Fremder Mann (1971)
/home/ahg/audio/rosenberg_fremdermann.mp3
#EXTINF:-1, Radio 1920: Musik der 1920er bis 40er
http://stream.laut.fm/radio1920
```

Die erste Zeile `#EXTM3U` kennzeichnet eine extended M3U playlist. Danach folgen jeweils zwei Zeilen für eine Audio- oder Videodatei oder die URL eines Internetsenders. Nach `#EXTINF:` wird für Audio- oder Videodateien die Länge des Stücks in Sekunden eingetragen. Diese Information kann man mit dem Programm `mediainfo` erhalten, siehe oben (Seite 197). Für Internetsender wird die Länge mit -1 angegeben. Dann folgen ein Komma und ein kurzer Text, der das Stück beschreibt (ohne Komma , und ohne Bindestrich -). Der Text darf mit einem Leerzeichen beginnen (zwecks Einrückung). In der Zeile danach wird der Pfad zur Datei angegeben.

Man kann Kommentarzeilen einfügen, die nur von Menschen, aber nicht vom Abspielprogramm gelesen werden sollen. Sie müssen mit einer Raute # beginnen und unmittelbar danach darf nicht die Zeichenkette `EXT` stehen.

Audio-Player MOC

Der Audio-Player MOC (Music on Console) spielt Audio-Dateien und playlists aus einem Verzeichnis. Auch Internetradiosender, die in einer playlist angegeben sind, werden abgespielt (siehe Seite 197). Unterstützte Audioformate sind unter anderem MP3, Ogg Vorbis, FLAC, Musepack (mpc), Speex, Opus, WAVE, WMA, RealAudio, AAC, MP4 und MIDI. Auch die Tonspur von Videodateien (Format MP4) wird wiedergeben.

MOC wird mit einer Konfigurationsdatei und Tastenbefehlen gesteuert. Er kann auch im Hintergrund laufen. Nach der Installation mit

```
sudo apt install moc moc-ffmpeg-plugin
```

wird das versteckte Verzeichnis `moc` erzeugt

```
mkdir .moc
```

und die Konfigurationsdatei namens `config` geschrieben.

```
nano /home/ahg/.moc/config
```

Wenn unsere Audiodateien im Verzeichnis `/home/ahg/audio` liegen und wir den HDMI Audioausgang benutzen, schreiben wir folgende Zeilen:

```
# MOC Konfiguration
MusicDir = "/home/ahg/audio"
StartInMusicDir = yes
Sort = FileName
Repeat = yes
Shuffle = yes
AutoNext = yes
SoundDriver = ALSA
ALSADevice = default
ALSAMixer1 = HDMI
ALSAMixer2 = Headphone
Allow24bitOutput = yes
```

Wir speichern dies (`Ctrl-O Y`) und verlassen den Editor `nano` (`Ctrl-X`).

Der Wert der Option `ALSADevice` bestimmt die ALSA-Soundkarte. Mögliche Werte sind `default` (die Standard-ALSA-Soundkarte, hier: HDMI), `hw:0` (ALSA-Soundkarte 0, hier: HDMI) und `hw:1` (ALSA-Soundkarte 1, hier: der analoge Audioausgang). Mit `ALSAMixer1` und `ALSAMixer2` werden verfügbare ALSAMixer angegeben.

Soll die analoge Audioausgabe benutzt werden, ändert man die Zeile `ALSADevice = default` in `ALSADevice = hw:1`.

Mit

```
mocp
```

wird der Audioplayer gestartet und nach Auswahl einer Audiodatei (mit den Pfeiltasten und der ENTER-Taste) spielt die Musik. Wichtige Tastenbefehle sind

h Anzeige einer Liste der Tastenbefehle ein / aus
q (Q ohne ⇑) zurück zur Kommandozeile, MOC läuft im Hintergrund
Q (⇑-Q) beendet den MOC
↑ in der Audioliste nach oben
↓ in der Audioliste nach unten
TAB (Tabulator) zwischen playlist und Dateiliste umschalten
C (⇑-C) Clear playlist (Löschen der aktuellen playlist, Dateien bleiben erhalten)
< leiser
> lauter
S (⇑-S) Shuffle (zufällige Wiedergabe) ein / aus
<space> (Leertaste) Pause ein / aus

Dabei ist ⇑ die Umschalttaste (Shift). Wie bei Audio-Playern üblich, gibt es unzählige weitere Möglichkeiten zur Einstellung und Steuerung.

Nach dem Verlassen der MOC Schnittstelle mit `q` (klein geschriebenes q) läuft MOC im Hintergrund (unsichtbar) weiter. In der Kommandozeile öffnet der Befehl

```
mocp
```

die `moc` Schnittstelle wieder, während

```
mocp -x
```

MOC beendet.

Audio-CDs rippen mit abcde

Rippen ist die Übertragung von Daten auf ein anderes Speichermedium, wobei das Datenformat geändert werden kann. Ein Programm zum Rippen von Audio-CDs (CD-Ripper) ist `abcde` („a better cd encoder“), das mit

```
sudo apt install abcde eyed3 lame
```

installiert wird.

Einige Einstellungen sollte man in einer Konfigurationsdatei vornehmen. Dazu kopiert man die allgemeine Konfigurationsdatei in das Heim-Verzeichnis des Nutzers `pi`

```
cp /etc/abcde.conf /home/ahg/.abcde.conf
```

Änderungen in der Datei `/home/ahg/.abcde.conf` kann man mit einem Texteditor wie `nano` vornehmen. Zunächst sind alle Zeilen durch das Zeichen `#` auskommentiert und es gelten die Standardeinstellungen. Man muss also nicht alle Zeilen ändern.

Nach dem Einlesen der allgemeinen Information aus der CD sucht `abcde` (mit Internetverbindung) in einer öffentlichen Datenbank nach Information zur CD. Die früher genutzte Datenbank FreeDB gibt es aber nicht mehr und wurde durch GnuDB ersetzt.

Wir ersetzen die Zeile `#CDDBURL="http://freedb.freedb.org/~cddb/cddb.cgi"` durch

```
CDDBURL="http://gnudb.gnudb.org/~cddb/cddb.cgi"
```

falls erforderlich. Wenn man eine neue Version von `abcde` hat, ist möglicherweise bereits die neue Datenbank eingetragen. Die Zeile `#CDDBSUBMIT=freedb-submit@freedb.org` ersetzen wir durch

```
CDDBSUBMIT=submit@gnudb.org
```

falls erforderlich.

Die einzelnen Tracks (Stücke auf der CD), und damit die erzeugten Audiodateien, werden von `abcde` automatisch nummeriert. Um die Sortierung bei einer späteren Auflistung der Audiodateien auch bei einer zweistelligen Zahl von Tracks richtig zu machen, ersetzen wir die Zeile `#PADTRACKS=n`.

```
PADTRACKS=y
```

Die Zeile `#OUTPUTDIR=‘pwd‘` ersetzen wir, um das Zielverzeichnis der erzeugten Audiodateien festzulegen.

```
OUTPUTDIR=/home/ahg/audio/
```

Die Zeile `#OUTPUTTYPE=ogg` ersetzen wir, so dass die Audiodateien im gewünschten Format (zum Beispiel `mp3`, `ogg` oder `wav`) erzeugt werden. Für MP3 schreiben wir

```
OUTPUTTYPE=mp3
```

Standardmäßig enthält der von `abcde` gebildete Name einer Audiodatei einen Punkt nach der Track-Nummer. Ich empfehle, diesen Punkt in der Zeile
`#OUTPUTFORMAT='${ARTISTFILE}-${ALBUMFILE}/${TRACKNUM}.${TRACKFILE}'`
wegzulassen und stattdessen folgendes zu schreiben:

```
OUTPUTFORMAT='${ARTISTFILE}-${ALBUMFILE}/${TRACKNUM}${TRACKFILE}'
```

Die Zeile `#LOWDISK=n` kann man ersetzen, wenn eine SD-Karte mit wenig freiem Speicher benutzt wird. Mit großem Speicher belässt man den Standardwert `LOWDISK=n`.

```
LOWDISK=y
```

Ist ein CD-Abspielgerät über USB angeschlossen und eine Audio-CD eingelegt kann man mit dem Befehl

```
abcde 1
```

Track 1 auslesen und als Audiodatei speichern.

Es kann sein, dass die Datenbank GnuDB keine Information zur CD hat. Andererseits kann mehr als ein Eintrag vorliegen. Auf die Frage `Which entry would you like abcde to use ...` kann man 0 antworten, wenn kein Datenbankeintrag benutzt werden soll, aber man kann auch zum Beispiel 1 (für den ersten Datenbankeintrag) wählen.

Auf die Frage `Edit selected CDDB data [Y/n]` kann man einfach die ENTER-Taste drücken und wählt damit den Standardwert `Y` = Ja. Es öffnet sich unser Standard-Editor (zum Beispiel `nano`) und wir machen zum Beispiel folgende Angaben

```
DTITLE=rosenberg / cd_1
DGENRE=Pop
TTITLE0=er_ist_nicht_wie_du
```

und lassen die anderen Zeilen unverändert. Man sollte in den Zeichenketten keine Doppelpunkte verwenden, da sie Teil des Verzeichnisnamens beziehungsweise des Dateinamens werden und Doppelpunkte in Dateinamen zu Problemen führen können. Leerzeichen sollte man ebenfalls nicht verwenden, außer zwischen „artist“ (`rosenberg`) und „album“ (`cd_1`). Dann verlassen wir den Editor in üblicher Weise. Auf die folgende Frage `Is the CD multi-artist [y/N]` kann man ebenfalls einfach die ENTER-Taste drücken und wählt damit den Standardwert `N` = Nein.

Nun beginnt das Auslesen der CD, das Encoding im gewählten Audio-Format und die Speicherung der Audiodatei. Im Verzeichnis `rosenberg-cd_1` finden wir anschließend die Audiodatei `1.er_ist_nicht_wie_du.mp3`.

Will man die ganze CD rippen, gibt man den folgenden einfachen Befehl ein:

```
abcde
```

Will man Tracks 1, 2, 6,7,8 der CD rippen, gibt man den folgenden Befehl ein:

```
abcde 1 2 6-8
```

yt-dlp

Mit `yt-dlp` kann man Videodateien (oder nur deren Audiospur) von Mediatheken oder Videoportalen wie YouTube herunterladen.

Die Installation über die Paketverwaltung liefert meistens eine veraltete Version, die vielleicht nicht mehr funktioniert. Daher ist eine Installation von der GitHub Webseite `https://github.com/yt-dlp/yt-dlp` besser. Wir brauchen dafür zwei, jeweils einzeilige Befehle. Der erste ist

```
sudo wget https://github.com/yt-dlp/yt-dlp/releases/latest/down
load/yt-dlp -O /usr/local/bin/yt-dlp
```

wobei der Befehl hier drucktechnisch in zwei Zeilen wiedergegeben wird. Dr zweite ist

```
sudo chmod a+rx /usr/local/bin/yt-dlp
```

Wen eine Aktualisierung (update) notwendig ist, kann sie mit dem Befehl

```
sudo yt-dlp -U
```

geschehen.

Wenn man, um Speicherplatz zu sparen, ein Video von YouTube (Mina - Heißer Sand, 1962) in der schlechtesten Qualität herunterladen will, geht das zum Beispiel so:

```
yt-dlp -o "mhs.mp4" -f worst https://www.youtube.com/watch?v=B13iJJdrbZw
```

wobei das Ergebnis in der Datei `mhs.mp4` abgespeichert wird und anschließend mit

```
mplayer mhs.mp4
```

angeschaut werden kann, wenn der MPlayer installiert ist (sihe Seite 186). Eine bessere Steuerung der Videoqualität, erreicht man mit der Option `-S`. Zun Beispiel wird mit

```
yt-dlp -o "mhs.mp4" -S "+res:360,codec:h264" url
```

ein Video mit der URL url in der besten Auflösung, aber höchstens 360p heruntergeladen. Falls es kein solches gibt, wird das Video mit der geringsten Auflösung heruntergeladen. Gibt es mehrere Videos, die dieser Bedingung genügen, wird (wenn möglich) das mit dem Codec h264 genommen. Das Video wird in der Datei `mhs.mp4` gespeichert.

Mithilfe von `ffmpeg` kann man eine Videodatei (Connie Francis - Heißer Sand, 1966) nachträglich in eine Audio-Datei umwandeln (siehe Seite 197). Zum Beispiel wird mit

```
yt-dlp -o "cfhs.mp3" -x --audio-format mp3
https://www.youtube.com/watch?v=EaOB_jOS7Cc
```

ein Musikvideo heruntergeladen, in eine Audiodatei umgewandelt und unter dem Namen `cfhs.mp3` gespeichert. Beachte: Der einzeilige Befehl wurde hier drucktechnisch in zwei Zeilen wiedergegeben.

Die Videos von YouTube sind in verschiedenen Formaten verfügbar. Darunter sind reine Videoformate (ohne Ton) und reine Audioformate (sowie Dateiformate mit Einzelbild). Mit diesen werden in modernen Echtzeit-Übertragungen Audio- und Videodaten getrennt übertragen und erst im Ziel vom Medienabspielgerät wieder zusammengefügt. Es ist daher meistens möglich, nur die Audiodatei eines YouTube-Videos herunterzuladen. Für eine schnelle Übertragung kann man die Audio-Qualität gering halten. Mit

```
yt-dlp -f worstaudio https://www.youtube.com/watch?v=Gnmq27MCkFY
```

bekommt man zum Beispiel von einem Musikvideo nur eine Datei mit der Endung `.webm`, welche Audiodaten im Format Opus, sowie Zusatzinformationen enthält. Das Format Opus wird von den meisten Medienspielern abgespielt, auch wenn es in der WEBM-Datei verpackt ist. Man kann die Musik also zum Beispiel mit dem MPlayer anhören. Oft möchte man das beste Audioformat ohne Video herunterladen. Das geschieht mit

```
yt-dlp -f bestaudio https://www.youtube.com/watch?v=Gnmq27MCkFY
```

wenn ein reines Audioformat verfügbar ist.

Man kann die beim Server verfügbaren Datei-Formate für eine URL `url` mit

```
yt-dlp -F url | less
```

abrufen. Wenn es zum Beispiel ein Format mit der Nummer `17` gibt, lädt der Befehl

```
yt-dlp -f 17 url
```

die Datei im entsprechenden Format herunter.

Manche Videos können Untertitel in verschiedenen Sprachen zur Verfügung stellen.

```
yt-dlp --list-subs url
```

listet die verfügbaren Untertitel (subtitles) des YouTube-Videos mit der URL `url` auf, falls es welche gibt. Der Befehl

```
yt-dlp --write-sub --sub-lang de --skip-download url
```

speichert die deutschen Untertitel des YouTube-Videos mit der URL `url` (zum Beispiel `https://www.youtube.com/watch?v=i93Z7zljQ7I`) im Format WebVTT (Datei-Endung `.vtt`) ab. Die Option `-skip-download` bewirkt, dass die eigentliche Videodatei nicht heruntergeladen wird. Eine WebVTT-Datei ist eine Textdatei, in der die Untertitel in einem bestimmten Format angeordnet sind. Sie kann mit jedem Texteditor gelesen (und geschrieben) werden. Die Untertitel in der WebVTT-Datei kann man leicht ohne die besondere Formatierung in eine Textdatei schreiben, siehe Seite 273.

Gibt es Untertitel, kann man das Video mit den Untertiteln im Bild herunterladen:

```
yt-dlp -f 17 --embed-subs url
```

streamlink

Das Programm `streamlink` vermittelt zwischen Webseiten, die einen Video-Stream anbieten, und einem Video-Player, welcher den Stream abspielt. Man könnte es zwar über die Paketverwaltung installieren, aber dann würde man eine alte Version erhalten, die nicht den Formatänderungen mancher Anbieter gefolgt ist. Für eine aktuelle Installation:

```
pip3 install --user --upgrade streamlink
```

Damit die Shell die so installierten Dateien findet, muss die Umgebungsvariable `PATH` ergänzt worden sein, wie in Abschnitt 2.1.3 auf Seite 50 beschrieben. Falls nach einiger Zeit Probleme bei der Nutzung von `streamlink` entstehen, kann es helfen, wenn man das Programm durch erneute Ausführung des obigen Installationsbefehls aktualisiert.

Um einen Video-Stream lesen zu können, braucht `streamlink` ein passendes Plug-in. Es verfügt über eine Menge von eingebauten Plug-ins (built-in plugins). Die Liste aller verfügbarer Plug-ins wird mit dem Befehl

```
streamlink --plugins
```

angezeigt.

Mit `streamlink` und einem Mediaplayer (in der Regel VLC) kann man einen Videostream (oder Audiostream) sofort abspielen oder ihn erst speichern. Insbesondere bei älteren Modellen des Raspberry Pi ist es besser, ein Video zunächst herunterzuladen und danach die gespeicherte Datei abzuspielen.

Meistens werden von einem Video-Titel mehrere streams mit unterschiedlicher Auflösung angeboten. Die höchste Auflösung verbraucht am meisten Speicherplatz und wird durch die Option `best` ausgewählt. Für unseren Raspberry Pi wollen wir uns oft mit der niedrigsten Auflösung begnügen und wählen daher die Option `worst`.

Wir schauen uns zunächst die Speicherung am Beispiel eines YouTube-Videos an. Um die Eingabe des richtigen Befehls zu vereinfachen, schreiben wir ein Shellscript dafür

```
yta="https://www.youtube.com/watch?v="
ytb="Y2qYvVbAc3Y"
url=$yta$ytb
dir="/home/ahg/video/"
nam="huesch_hildebrandt.mp4"
out=$dir$nam
qal="worst" # best / worst
streamlink -o $out $url $qal
```

und speichern es in der Datei `strili`. Die URL wurde aus dem unveränderlichen Anfangsteil `yta` und dem veränderlichen (vom YouTube-Video abhängigen) Endteil `ytb` zusammengesetzt. Ebenso besteht der Pfad zur Speicherung aus dem Verzeichnis `dir` und dem Namen der Datei `nam`, der für das Video gewählt wird. Mit dem Befehl

```
chmod a+x strili
```

wird die Datei ausführbar gemacht und kann dann einfach mit

```
strili
```

ausgeführt werden (siehe Abschnitt 4.3 auf Seite 136). Nach der Speicherung kann man das Format (mithilfe der `ffmpeg`-Bibliothek) noch ändern. Alternativen zur Speicherung mit `streamlink` sind `youtube-dl` (siehe Seite 201) und VLC.

Es gibt zahlreiche Mediatheken, deren Videos von `streamlink` gelesen werden können (siehe *Mediatheken* im Abschnitt 9.2 auf Seite 417). Im folgenden Beispiel geht es um ein Video der ARD Mediathek und wir prüfen zunächst, welche Auflösungen zur Verfügung stehen. Dazu wird `streamlink` mit der URL des Videos ohne weitere Optionen aufgerufen. Wir passen die Datei `strili` entsprechend an:

```
url1="https://www.ardmediathek.de/video/planet-schu"
url2="le-geschichte/wie-bauten-die-roemer/swr/Y3JpZD"
url3="ovL3BsYW5ldC1zY2h1bGUuZGUvQVJEXzg3MzJfdmlkZW8/"
url=$url1$url2$url3
dir="/home/ahg/video/"
nam="wie_roemer_bauten.mp4"
out=$dir$nam
qal="worst" # best / worst
streamlink $url
# streamlink -o $out $url $qal
```

Da die URL des Videos sehr lang ist, wird sie hier in vier Zeilen durch Konkatenation (Zusammenfügung von Zeichenketten) gebildet. In der Antwort von `streamlink` erfahren wir, dass es ein passendes Plug-in gibt und welche streams verfügbar (available) sind, hier: `78k (worst)`, `396p` und `720p (best)`. Da `78k (worst)` nur eine Tonspur ohne Bild liefert, werden wir `396p` wählen. Wir ändern die Datei `strili` in

```
url1="https://www.ardmediathek.de/video/planet-schu"
url2="le-geschichte/wie-bauten-die-roemer/swr/Y3JpZD"
url3="ovL3BsYW5ldC1zY2h1bGUuZGUvQVJEXzg3MzJfdmlkZW8/"
url=$url1$url2$url3
dir="/home/ahg/video/"
nam="wie_roemer_bauten.mp4"
out=$dir$nam
qal="396p" # best / worst
# streamlink $url
streamlink -o $out $url $qal
```

uns speichern damit das gewünschte Video im angegeben Verzeichnis.

Man kann eine Konfigurationsdatei namens `.streamlinkrc` im home-Verzeichnis des Nutzers, `/home/ahg/`, anlegen und dort Voreinstellungen vornehmen, aber wenn man `streamlink` über ein Shellscript aufruft, ist das überflüssig.

DVDs

Will man DVDs decodieren, braucht man die Bibliothek `libdvdcss2`. Die Installation erfordert einen Umweg. Zuerst installiert man das Paket `libdvd-pkg`. Dann gibt man

```
sudo dpkg-reconfigure libdvd-pkg
```

ein. Die Frage nach „automatic upgrades for libdvdcss2“ kann man verneinen (`No`), wenn man keine automatische Aktualisierung der Bibliothek wünscht. In diesem Fall führt obiger Befehl zu einer (manuell ausgelösten) Aktualisierung.

Grundsätzlich kann man DVDs mithilfe des `mplayer` abspielen. Der Befehl

```
mplayer -fs -vo fbdev dvd://
```

genügt. Allerdings wird das Abspiel durch die fehlende „Hardwarebeschleunigung“ erschwert. Hat man das Pakett `dvdbackup` installiert, erhält man mit dem Befehl

```
dvdbackup -I
```

Information zum Inhalt der DVD.

6.2 Internet

6.2.1 HTTP requests und Browser

In der Kommandozeile gibt es verschiedene Programme für HTTP-Abfragen und das Herunterladen (Download) von Dateien. HTTP-Abfragen, englisch HTTP requests, werden später genauer erklärt, siehe Seite 341. Die Programme `curl` und `wget` wurden im Abschnitt „Netzwerk“ ab Seite 115 kurz eingeführt. Wir werden gleich auf ähnliche Programme, nämlich `axel` und HTTPie eingehen.

Die meisten Webseiten im *World Wide Web* (www) sind mit graphischen Elementen und besonderen Formatierungen versehen und können vollständig nur mit umfangreichen Browsern betrachtet werden, die eine graphische Oberfläche benötigen. Mit Linux ohne X sind wir in diesem Bereich beschränkt. Immerhin, einiges ist möglich.

Schnelle Downloads mit axel

Der Download-Manager `axel` lädt Dateien mit den Internet-Protokollen FTP, FTPS, HTTPS und HTTPS ziemlich schnell herunter. Er kann mit dem Befehl

```
sudo apt install axel
```

installiert werden. Danach kann zum Beispiel mit dem Befehl

```
axel -c http://www.archive.org/download/\
winnetou1_librivox/winnetou1_librivox_64kb_mp3.zip
```

eine 481,4 MB große Datei heruntergeladen. Die Option `-c` bewirkt, dass ein Download nicht stattfindet, wenn es die Datei bereits gibt. Man beachte, dass in obigem Befehl ein Zeilenumbruch mit dem Zeichen \ eingefügt wurde, damit die Zeile nicht die Druckbreite dieses Manuskripts überschreitet. Während des Downloads werden der Fortschritt in Prozent, die Geschwindigkeit der Datenübertragung und die geschätzte Zeit bis zum Abschluss angezeigt. Obiges Beispiel liefert uns ein Hörbuch von LibriVox mit 27 Kapiteln in verlustfrei komprimierter Form (ZIP-Dateiformat). Das Entpacken durch

```
unzip winnetou1_librivox_64kb_mp3.zip
```

ergibt 27 Audiodateien im MP3-Format mit Karl Mays Buch Winnetou 1, welche mit dem MPlayer (siehe Seite 186) gehört werden können.

HTTPie für HTTP-Abfragen (HTTP requests)

Das Programm HTTPie für HTTP-Abfragen und das Herunterladen von Dateien aus dem Internet soll einfacher anzuwenden sein als die üblichen Programme `curl` und `wget`. HTTPie wird mit dem Befehl

```
sudo apt install httpie
```

installiert. Es wird mit dem Befehl `http` aufgerufen und erwartet einen URL (Internetadresse) als Argument. Vor der URL kann man eine HTTP Methode (`GET`, `POST`, `PUT`, `DELETE`, ...) angeben. Wird keine angegeben, dann wird `POST` verwendet, wenn Daten gesendet werden und andernfalls `GET`. Zwischen dem Befehlswort `http` und dem Methodennamen können Optionen stehen. Nach dem URL können Daten (request items) folgen, die gesendet werden sollen. Diese Daten bestehen aus einer Folge von Zeichenketten, die durch Leerzeichen getrennt sind, wobei jede Zeichenkette ein Paar aus key und value ist, das durch einen separator getrennt wird. Der separator bestimmt den Typ des Datenpaars. Die Syntax (Schreibweise) des Befehls hat also die Form

`http` Optionen Methode URL Datenpaare

Genaueres zur Nutzung erfährt man mit dem Befehl

```
http --help | less
```

Lynx

Ein textbasierter Webbrowser, der allerdings kein JavaScript verarbeitet und keine Bilder innerhalb des Texts zeigt, ist *Lynx*. Er wird mit

```
sudo apt-get install lynx lynx-common
```

installiert und mit

```
lynx
```

gestartet. Durch Drücken der Taste `q` und Bestätigung mit `y` oder `y` verlässt man den Browser. Eine englische Anleitung erhält man nach der Installation durch den Befehl

```
lynx /usr/share/doc/lynx/lynx_help/Lynx_users_guide.html.gz
```

Obwohl `lynx` keine Bilder innerhalb des Texts zeigt, kann er doch Verweise auf die Bilddateien ausgeben und diese kann man einzeln über den Framebuffer anzeigen lassen. Dafür genügt →, siehe unten. Mit Bildverweisen wird `lynx` durch

```
lynx -image_links
```

gestartet. Alternativ kann man eine Konfigurationsdatei ändern (siehe unten).

`lynx` kann mit einzelnen Tasten bedient werden:

g Eingabe einer URL (zum Beispiel `https://de-wiki.tk`)
\ Umschaltung zwischen „gerenderter“ Webseite und dem Quelltext der Seite
Bild ↓ in der Webseite zur nächsten Seite (page) weiter rücken
Bild ↑ in der Webseite zur vorigen Seite (page) weiter rücken
↓ Cursor zum nächsten Verweis (Link) bewegen
↑ Cursor zum vorigen Verweis (Link) bewegen
→ dem aktuellen Verweis (Link) folgen
← zur vorher gewählten Webseite zurück gehen
***** Anzeige der Bildverweise ein- oder ausschalten
q Beenden von `lynx`

Nach der Eingabe einer URL wird man zur Verwendung von Cookies gefragt. `A` erlaubt Cookies für die Webseite. Manchmal muss man die Zustimmung mehrfach erteilen. Es kommt vor, dass `lynx` „einfriert“ und man das Betriebssystem neu starten muss.

Moderne Webseiten enthalten meistens Graphik und anderen Schnick und Schnack, sodass `lynx` wohl nur in besonderen Fällen sinnvoll ist. Immerhin kann man das eine oder andere nachschlagen, zum Beispiel über Shellscripts mit bash:

```
lynx https://linuxconfig.org/bash-scripting-tutorial
```

Will man nur den Header der Internetseite anzeigen, geht das im Beispiel mit

```
lynx -head -dump https://linuxconfig.org/bash-scripting-tutorial
```

Allerdings hat `lynx` auch Vorteile: er ist sehr schnell und kann genau konfiguriert werden. `lynx` hat mehrere Konfigurationsdateien.

Wir betrachten hier nur Anpassungen in der Datei `/etc/lynx/lynx.cfg`, die mit einem Editor (als Administrator) vorgenommen werden können.

Die voreingestellte Startseite von `lynx` ist eine lokale Datei, die über `lynx` informiert. Um sie zum Beispiel durch eine Seite der Suchmaschine DuckDuckGo zu ersetzen, löscht man die Zeile

```
STARTFILE:file://localhost/usr/share/doc/lynx/lynx_help/about_lynx.html.gz
```

und schreibt unter der bereits vorhandenen Zeile

```
#STARTFILE:https://lynx.invisible-island.net/
```

die neue Zeile

```
STARTFILE:https://duckduckgo.com/lite/ # geaendert am ...
```

wobei ... im Kommentar für das Datum unserer Änderung steht.

Normalerweise fragt `lynx` den Nutzer, ob es einen Cookie des Servers annehmen darf. Um automatisch alle Cookies anzunehmen, ersetzen wir die Zeile

```
#ACCEPT_ALL_COOKIES:FALSE
```

durch

```
ACCEPT_ALL_COOKIES:TRUE # geaendert am ...
```

wobei ... im Kommentar für das Datum unserer Änderung steht.

Um Verweise auf eingebettete Bilddateien anzeigen zu lassen, ersetzen wir die Zeile

```
#MAKE_LINKS_FOR_ALL_IMAGES:FALSE
```

durch

```
MAKE_LINKS_FOR_ALL_IMAGES:TRUE # geaendert am ...
```

wobei ... im Kommentar für das Datum unserer Änderung steht.

Webseiten, die man mit textbasierten Browsern wie `lynx` brauchbar darstellen kann, findet man im Anhang (siehe Seite 411). Außerdem werden die Gopher-Seiten (siehe Seite 213) gut dargestellt.

Browser w3m

Der schlanke Browser `w3m` kann, nach Installation durch

```
sudo apt-get install w3m-img
```

Webseiten mit Bildern anzeigen, indem der Framebuffer statt X genutzt wird. Die Bedienung ist einfach. Zum Beispiel wird durch `w3m google.de` eine Suchmaschine aufgerufen. Man kann natürlich die URL einer Webseite angeben. Zum Beispiel wird mit

```
w3m "https://www.qwant.com?l=de&q=Wilhelm+Raabe"
```

die Ergebnisseite der Suchmaschine Qwant (Motto: „die Suchmaschine, die nichts über Sie weiß!“) für deutsche Webseiten mit dem Begriff „Wilhelm Raabe“ angezeigt.

`w3m` beherrscht nicht *JavaScript* und man kann daher Webseiten wie *Google Maps* nicht verwenden. Eine deutsche Anleitung erhält man nach der Installation durch

```
w3m /usr/share/doc/w3m/de/MANUAL.html
```

Durch Drücken der Taste `q` und Bestätigung mit `y` verlässt man den Browser. Mehr Information erhält man entsprechend mit der Datei `/usr/share/doc/w3m/de/FAQ.html`.

Die Farbgebung des Browsers ist zunächst nicht gut, denn blau gefärbte Verweise (Links) heben sich schlecht vom schwarzen Hintergrund ab. Wird `w3m` mit der Option `-M` (monochrome display) gestartet, ist die Farbgebung weniger aufdringlich.[1]

Die Farbgebung kann man auch im Einstellungsmenü (Option Setting Panel) ändern: Mit der Taste `o` (kleines o) öffnet man das Menü und mit der Tastenkombination `Shift-B` (großes B) könnte man es ohne Speicherung von Änderungen wieder verlassen.

Im Einstellungsmenü kann man sich mit dem Abwärtspfeil ↓ nach unten und mit dem Aufwärtspfeil ↑ nach oben bewegen. Unter `Farbeinstellungen` bewegt man sich zur Zeile `In Farbe anzeigen`, drückt die Tabulatortaste und aktiviert danach durch Drücken der ENTER-Taste die Auswahl `YES` (falls sie noch nicht aktiv ist). In der folgenden Zeile `Farbe für normalen Text` öffnet man rechts mit der ENTER-Taste das Auswahlfeld und wählt `schwarz`. Weiter unten in der Zeile `Hintergrundarbe` wählt man in gleicher Weise `weiß`. Dann bewegt man den Cursor zum Feld `[OK]` und bestätigt die Änderungen durch Drücken der ENTER-Taste. Man kann natürlich auch andere Farben wählen.

Innerhalb der Google-Webseite bewegt man den Cursor mit Pfeiltasten oder springt mit der Tabulatortaste zum nächsten Verweis. Im Eingabefeld für Suchbegriffe (im Leerraum zwischen zwei eckigen Klammern) drückt man die ENTER-Taste und kann dann einen Suchbegriff schreiben, zum Beispiel `Jena`. Die Eingabe wird mit der ENTER-Taste abgeschlossen und mit einem Tabulatorsprung gelangt man zum Verweis `Google-Suche`, den man mit der ENTER-Taste aktiviert. Auf der Ergebnisseite wählen wir den Verweis `Jena Lichtstadt: Startseite ...` und die Webseite der Stadt Jena wird geladen. Wir bewegen den Cursor nach unten und sehen schließlich viele Bilder. Der Oberbürgermeister lächelt uns freundlich an. Drücken von `Shift-B` bringt uns schnell zurück zur

[1] De gustibus et coloribus non est disputandum. Über Geschmack und Farben kann man nicht streiten.

vorher aufgerufenen Webseite. Mit der Taste `q` und Bestätigung mit `y` verlassen wir den Browser.

Die Nachfrage („Do you want to exit w3m?“, welche nach Drücken der Taste `q` zum Verlassen von `w3m` erscheint, kann man im Einstellungsmenü ausschalten. Dafür aktiviert man im Bereich `Weitere Einstellungen` beim Stichpunkt `Das Programm erst nach Bestätigung verlassen` die Auswahl `NO` statt `YES`. Dann bewegt man den Cursor zum Feld `[OK]` und bestätigt die Änderungen durch Drücken der ENTER-Taste.

Eine Datei, auf die ein Verweis zeigt, kann automatisch heruntergeladen und im Verzeichnis `/home/ahg` gespeichert werden, indem man die Taste `a` drückt und danach die ENTER-Taste.

Man kann `w3m` als Standard-Browser festlegen, wie auf Seite 51 beschrieben. Beim Aufruf von `w3m` ohne Angabe einer Webseite sollte eine voreingestellte Webseite, zum Beispiel die der Suchmaschine DuckDuckGo, geöffnet werden. Dies erreicht man durch einen Eintrag in der Datei `/home/ahg/.bashrc`, wie auf Seite 51 beschrieben.

Browser links2

Der Browser `links2` kann in einem Graphikmodus arbeiten. Leider verarbeitet er kein JavaScript. Man muss auch das Paket `gpm` zur Mausunterstützung installieren (siehe Seite 48), obwohl man `links2` ohne Maus bedienen kann.

```
sudo apt install gpm links2
```

`links2` arbeitet mit dem Framebuffer (driver fb) im Graphikmodus, wenn es mit der Option `-g` aufgerufen wird. Zum Beispiel lädt man die Suchmaschine DuckDuckGo mit

```
links2 -g https://duckduckgo.com/lite/
```

Man kann mit Pfeiltasten (oder mit der Maus) navigieren. Mit der Escape-Taste ESC wird die Menüzeile auf- und zugemacht (oder man klickt mit der Maus auf die obere, weiße Zeile). In Menü kann man mit Pfeiltasten und ENTER Einstellungen vornehmen.

Unter `Setup` (`Einstellungen`) und `Language` (`Sprache`) wählt man `German` (`Deutsch`) für ein Menü in deutscher Sprache und mit `Save options` (`Optionen speichern`) werden die Einstellungen dauerhaft gespeichert. Dafür wird automatisch die Konfigurationsdatei des Nutzers `~/.links2/links.cfg` angelegt. Diese Konfigurationsdatei soll man nicht direkt (mit einem Editor) ändern, sondern nur indirekt über das Menü.

`links2` kann mit einzelnen Tasten bedient werden; einige wichtige sind:

\	Umschaltung zwischen „gerenderter“ Webseite und dem Quelltext der Seite
z	zurück (im Graphik-Modus)
x	vorwärts
→	dem aktuellen Verweis (Link) folgen
d	Speichern der Webseite in einer Datei
q	Beenden von `links2`

6.2.2 Internet-Geschwindigkeit, Internet-Suche und Gopher

Internet-Geschwindigkeitsmessung mit speedtest-cli

Nach Installation des Pakets `speedtest-cli` kann man mit

```
speedtest-cli
```

die aktuelle Geschwindigkeit der Internetverbindung messen.

DuckDuckGo-Suche mit ddgr

Das Programm `ddgr` leitet eine Suchanfrage an die Internet-Suchmaschine DuckDuckGo und gibt das Ergebnis auf dem Bildschirm aus. `ddgr` wird wie üblich über die Paketverwaltung installiert. Eine Suchanfrage, wie

```
ddgr Jena
```

liefert, ohne Optionen, 10 Suchergebnisse, oft hauptsächlich englischsprachige, ohne Begrenzung des Zeitraums der Veröffentlichung der Webseiten. Wichtige Optionen sind

`-n` *x* Anzahl der Suchergebnisse auf x begrenzen (auch ohne Leerzeichen nach `-n`)
`-r` *x* Region x der gesuchten Webseiten (auch ohne Leerzeichen nach `-r`)
`-t` *x* Zeitraum x der gesuchten Webseiten (auch ohne Leerzeichen nach `-t`)

Eine Übersicht der Optionen beim Aufruf von `ddgr` erhält man mit dem Befehl

```
ddgr -h | less
```

Um alle Suchergebnisse auf dem Bildschirm sehen zu können, sollte die Anzahl der gezeigten Ergebnisse eingeschränkt werden, zum Beispiel durch `-n3`. Für Suchergebnisse aus Deutschland wählt man `-rde-de` und entsprechend `-rus-en` für englischsprachige aus den USA, `-ruk-en` für das Vereinigte Königreich und `-rcn-zh` für China. Das Alter der gesuchten Webseiten schränkt man ein, indem man den Zeitraum der Veröffentlichung vor dem aktuellen Datum festlegt, zum Beispiel durch `-ty` für Webseiten, die höchstens ein Jahr alt sind (y für year, also Jahr). Entsprechend `-td` für Tag, `-tw` für Woche und `-tm` für Monat.

Die Ergebnisseite schließt mit einem Prompt ab, wo man einen kurzen Befehl eingeben kann. Hat man einen Standard-Browser definiert (siehe Seite 51), kann durch Angabe der Indexzahl ein Eintrag in der Ergebnisliste mit dem Standard-Browser geöffnet werden. Für eine Übersicht der in der Ergebnisseite verfügbaren Kurzbefehle gibt man `?` ein (abgeschlossen mit der ENTER-Taste). Einige wichtige Kurzbefehle sind

? Anzeige der Kurzbefehle in der Ergebnisseite
n Laden von x weiteren Ergebnissen, wobei x die in Option `-n` genannte Zahl ist
p Laden von x vorherigen Ergebnissen, wobei x die in Option `-n` genannte Zahl ist
f Anzeige der ersten Ergebnis-Seite
1 Öffne mit Standardbrowser Eintrag 1 der Ergebnisseite (entsprechend für weitere)

***** Eingabe, die nicht Kurzbefehl ist (*) $\Longrightarrow$ neue Suche mit gleichen Optionen
q Beenden von `ddgr`, auch möglich durch zweimaliges Drücken der ENTER Taste

`ddgr` hat keine Konfigurationsdatei, aber man kann den Aufruf, einschließlich Optionen, mittels `alias` als Kurzbefehl definieren. Zum Beispiel bewirkt

```
alias s='ddgr -n3 -rde-de -ty'
```

dass mit `s Suchbegriff` und den oben erläuterten Optionen nach `Suchbegriff` gesucht wird. Wenn man den eben genannten Befehl in die Datei `.bashrc` einfügt (siehe Seite 52), ist der Kurzbefehl `s` immer verfügbar. Die gespeicherten Optionen können durch erneutes Setzen der Optionen im Kurzbefehl überschrieben werden.

Eine Anleitung in Englisch findet man auf den Webseiten
https://github.com/jarun/ddgr und
https://fossbytes.com/search-duckduckgo-from-terminal-ddgr/

Internet-Suche mit surfraw

Verschiedene Suchmaschinen und andere Webseiten können mit dem Programm surfraw in der Kommandozeile angesprochen werden. Die Ergebnisse werden in einem Browser, zum Beispiel `w3m` (siehe Seite 209) dargestellt. Ohne einen Browser, der JavaScript verarbeitet, sind wir leider in der Auswahl der Webseiten eingeschränkt. Eine Konfigurationsdatei wird in der Regel im Verzeichnis `/.config/surfraw/` angelegt, das man mit dem Befehl

```
mkdir ~/.config/surfraw
```

erzeugt. In diesem Verzeichnis wird die Konfigurationsdatei `conf` mit

```
touch ~/.config/surfraw/conf
```

erzeugt und kann danach mit einem Texteditor wie `nano` (siehe Seite 131) geschrieben werden. Zur Festlegung des Browsers `w3m` für das Programm `surfraw` und der Bitte um 10 Suchergebnisse (welche nicht alle Webseiten befolgen) schreiben wir die Zeilen

```
SURFRAW_text_browser=w3m
SURFRAW_results=10
```

in die Konfigurationsdatei `conf`. Nun können wir das Programm `surfraw` mit der Abkürzung `sr` aufrufen und geben als erstes Argument die Webseite an, die wir ansprechen wollen. Als zweites Argument können wir einen Begriff schreiben, welcher als Argument an die angesprochene Webseite übermittelt wird. Zum Beispiel wird durch

```
sr google "Schlacht bei Jena"
```

mit der Suchmaschine `google` nach der Schlacht bei Jena gesucht. Statt `google` kann man auch die Suchmaschinen `bing` oder `duckduckgo` verwenden. Beim Onlineversandhändler `amazon` kann man zum Beispiel mit

```
sr amazon -search=books -country=de -q "Raspberry Pi"
```

nach deutschen Büchern über den Raspberry Pi suchen. Die Informationsseiten ArchWiki sind nicht nur für Arch Linux, sondern auch für andere Linux-Varianten interessant. Mit

```
sr archwiki surfraw
```

erfährt man etwas über das Programm `surfraw`. Fragen kann man in Englisch an das Internetportal https://de.wikipedia.org/wiki/Ask.com senden, zum Beispiel

```
sr ask "Why is the moon white?"
```

und manchmal bekommt man Verweise auf gute Antworten. Die Herkunft eines englischen Worts kann man auf der Webseite `https://www.etymonline.com/` erfahren.

```
sr etym "catch 22"
```

erläutert den Begriff catch-22 .

Gopher: Hier endet das World Wide Web !

Das Internet ist mehr als das World Wide Web (WWW). In den 1990er Jahren war Gopher ein verbreitetes Netzwerkprotokoll. Mit ihm können insbesondere Textdokumente von Servern angeboten und von Clients abgerufen werden. Viele moderne Browser unterstützen das Gopher-Protokoll nicht, aber Lynx (siehe Seite 207) schon. Mit

```
lynx gopher://gopherspace.de
```

gelangen wir zu einem deutschsprachigen Gopher-Server.

Wir öffnen den Verweis `weiter zum Gopherspace` und erhalten ein kleines Menü. Wenn man den Verweis `Jounalist-Magazin` öffnet, kann man aktuelle Nachrichten lesen. Mit `q`, und zur `j` zur Bestätigung beendet man Lynx.

Deutschsprachige Nachrichten bietet auch das „grosse linke Nachrichten-Portal“ der *tageszeitung* aus Berlin (taz) im Gopherspace an:

```
lynx gopher://taz.de:70/1
```

Englischsprachige Nachrichten, als leichtgewichtiges CNN, gibt es hier:

```
lynx gopher://codevoid.de/1/cnn
```

~ubergeek bietet Verweise auf verschiedene englischsprachige Nachrichten an (Hacker News, Tilderverse News, Reuters Science News, Reuters Technology News, NPR News, CNN News, CS Monitor):

```
lynx gopher://tilde.team/1/~ubergeek/news/
```

Englische „information of historic interest“ bietet

```
lynx gopher://gopher.quux.org
```

Man kann mit Gopher auf die englische Wikipedia zugreifen:

```
lynx gopher://gopherpedia.com/1/
```

Ebenso gibt es eine Schnittstelle zu Reddit, einer Internetseite, auf der Nutzer aktuelle Texte (mehr oder weniger interessante Geschichten und Nachrichten) bereitstellen:

```
lynx gopher://gopherddit.com/1/
```

Insgesamt gibt es weltweit etwa 200 bis 400 aktive Gopher-Server. Eine Suchmaschine für Gopher-Server ist Veronica-2. Man erreicht sie mit folgendem Befehl:

```
lynx gopher://floodgap.com/1/v2
```

6.2.3 Wikipedia, Nachrichten, Wetter und Geschwätz

wikipedia2text

Das Paket `wikipedia2text`, das mit dem Befehl

```
sudo apt install wikipedia2text
```

installiert wird, ermöglicht es, Artikel der Wikipedia auf Englisch oder Deutsch in einem Terminal auszugeben.

```
wikipedia2text Empedocles | less
```

gibt einen englischen Wikipedia-Artikel über einen griechischen Philosophen aus. Die Übergabe an das Programm `less` bewirkt, dass wir uns mit den Cursor-Tasten ↓ und ↑ im langen Text bewegen können; `q` beendet es.

Mit der Option `-l de` wird eine deutsche Wikipedia-Seite aufgerufen. die Option `-s` beschränkt die Ausgabe auf die Zusammenfassung (englisch: summary). Der Befehl

```
wikipedia2text -s -l de Empedokles
```

ergibt daher eine deutsche Zusammenfassung über Empedokles. Der Befehl

```
wikipedia2text -l de Empedokles | grep -m 1 http
```

gibt nur die URL der Seite aus.

wikicurses

Sehr schön kann man in der Wikipedia mit dem Programm `wikicurses` lesen. Voraussetzung ist, dass die Pakete `python3-pip` und `python3-lxml` bereits installiert wurden, siehe Seite 78. Installiert wird es mit

```
pip3 install wikicurses
```

Eventuell muss man mit `pip3` auch die Module `bs4`, `lxml` und `requests` installieren. Nun schreiben wir die Konfigurationsdatei.

```
sudo nano /etc/wikicurses.conf
```

Die Datei soll folgenden Inhalt haben:

```
# Konfigurationsdatei fuer wikicurses
[general]
default = Wikipedia
mouse = False
hide_references = False

[keymap]
q = quit
c = contents
o = open
h = back
l = forward
left = back
right = forward

[Wikipedia]
url = http://en.wikipedia.org/w/api.php
```

Nun können wir das Programm mit

```
wikicurses
```

starten und lesen, nach einigem Warten, die Eingangsseite der englischen Wikipedia. Mit der Taste `o` wird ein Rahmen geöffnet, in dem ein Suchbegriff eingeben werden kann, abgeschlossen mit der Enter-Taste. Nach einiger Zeit erscheint der entsprechende Artikel. Durch Drücken der Taste `q` beenden wir das Programm.

newsboat

Einen Nachrichtenüberblick oder kurze, schnell verfügbare Texte, kann man in einem Web-Feed ohne Browser mit einem Feedreader, zum Beispiel *Newsboat*, im Terminal lesen. Newsboat ist ein Nachfolger von Newsbeuter, welcher nicht mehr zur Verfügung steht (oder nur in sehr alter Version). Nach der Installation mit

```
sudo apt install newsboat
```

kann man das Programm mit dem Befehl

```
newsboat
```

starten. Zunächst erhält man eine Fehlermeldung, weil noch keine Web-Feeds in der Datei `/home/ahg/.newsboat/urls` (configuration file) eingetragen wurden. Man öffnet mit einem Texteditor eine neue Datei `urls` im Verzeichnis `/home/ahg/.newsboat`

```
nano /home/ahg/.newsboat/urls
```

und trägt die gewünschten Web-Feeds ein, zum Beispiel

```
https://physicsworld.com/feed
```

Mehr Web-Feeds sind im Anhang aufgelistet, siehe Seite 409. Nach dem Schließen der Datei kann man das Programm `newsboat` benutzen.

```
newsboat -r
```

Die Option `-r` bewirkt, dass alle Web-Feeds beim Start des Programms aktualisiert werden. Wenn man in der Datei `urls` sehr viele Web-Feeds eingetragen hat, die man selten nutzt, ist es jedoch besser, nach dem Start von `newsboat` gezielt nur die Web-Feeds zu aktualisieren, die man gerade lesen will.

Mit den Pfeiltasten und der ENTER-Taste wählt man einen Web-Feed aus und darin einen Nachrichteneintrag. Ein „N“ auf der linken Seite der Zeile für einen Eintrag bedeutet, dass der Eintrag noch nicht gelesen wurde. (Ein „N“ in Zeilen der Liste der Web-Feeds ist ohne praktische Bedeutung.) Die wichtigsten Tastenbefehle zur Steuerung von `newsboat` werden unten am Bildschirm angezeigt. Mit der Taste `o` wird der link (Verweis) im Browser geöffnet, wenn es im Eintrag einen link gibt. Als Browser wird der gewählt, der in der Umgebungsvariable `BROWSER` festgelegt ist (siehe Seite 51) oder `lynx`, falls es die Umgebungsvariable nicht gibt.

In der Datei `/home/ahg/.newsboat/urls` kann man Kommentarzeilen, die mit # beginnen, einfügen. Sie werden vom Programm nicht angezeigt.

Es ist oft gut, einem Web-Feed einen Namen zu geben, mit dem er in der Liste angezeigt wird. Wie das geht, zeigt folgendes Beispiel

```
https://www.ad-hoc-news.de/rss/boerse.xml "~Börsennachrichten (deutsch)"
```

Eine englische Anleitung zu Newsboat findet man auf der Webseite
https://newsboat.org/releases/2.26/docs/newsboat.html

Kurznachrichten via getnews.tech

Über getnews.tech kann man länderspezifisch sortierte Kurznachrichten für ein Terminal von News API erhalten. Allerdings sind die bereit gestellten Dokumente für ein Terminalfenster mit vertikalem Bildlauf (scrolling) formatiert. Da diese Funktion in der einfachen Linux-Konsole fehlt, rufen wir die Nachrichten einzeln ab. Mit dem Befehl

```
curl de.getnews.tech/n=1,p=1
```

erhalten wir die neueste deutsche Nachricht (Kürzel `de`). Für die vorletzte ersetzen wir `p=1` durch `p=2`, und so weiter, wobei die Pfeiltaste ↑ der `bash` history-Funktion hilft (siehe Seite 110). Die Erstellung eines Shellscripts (oder Pythonprogramms) zur Vereinfachung des Abrufs belassen wir als Übungsaufgabe für den geneigten Leser.

Wetterbericht

Den lokalen Wetterbericht, zum Beispiel für Jena (siehe Abbildung 6.1), erhält man mit

```
curl de.wttr.in/Jena?F2nqd
```

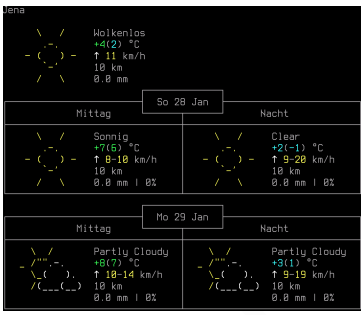


Abbildung 6.1: Wetter

Die graphiksche Darstellung besteht aus Zeichen. Das `de.` wählt die deutsche Sprache. Mit `F` wird ein abschließender Hinweis auf eine Webseite unterdrückt. Die `2` bewirkt, dass auch die Wettervorhersage für morgen gezeigt wird. Das `n` beschränkt die Zeiten auf Tag und Nacht und erzeugt eine schmalere Ausgabe auf dem Bildschirm. Das `q` unterdrückt am Ende die Ausgabe einer Zeile, in welcher der Ort mit seinen Koordinaten genannt wird. Das `d` beschränkt die Zeichenmenge auf den Font unserer Konsole, was sich auf die verfügbaren Pfeile für die Windrichtung auswirkt. Mit

```
curl wttr.in/:help | less
```

erfährt man die Bedeutung der Optionen.

Ein größerer Wetterbericht passt nicht auf einen Bildschirm. Man könnte ihn aber zunächst im Format PNG als Bild in der Datei `wetter` speichern

```
curl -o wetter http://wttr.in/Jena_lang=de.png
```

wobei ein Unterstrich (statt eines Fragezeichens) nach der Ortsbezeichnung (hier: `Jena`) steht. Mit `lang=de` erreichen wir eine deutsche Ausgabe. Danach wird das Bild durch

```
fbi -a wetter
```

mithilfe des Programms `fbi` angezeigt (siehe Seite 173). Die Option `-a` steht für „autozoom“ und bewirkt eine automatische Vergrößerung (zoom). Dies ist derzeit aber nicht möglich („PNG support temporary disabled).

Wenn man das Paket `ansiweather` installiert hat, erhält man mit dem Befehl

```
ansiweather -l Jena,DE -f 3
```

einen einzeiligen Wetterbericht für Jena.

Reddit

Die weltweit beliebte Webseite für Diskussionen und Bilder der Nutzer über alle möglichen Themen, Reddit, kann man mit dem Programm `tuir`, das mit

```
pip3 install --user tuir
```

installiert wird, besuchen. Die Vielzahl von Beiträgen sind in Teilmengen sortiert, welche subreddits heißen. Einige Beispiele:

- AskScience – wisenheimers' answers to foolish questions on science
- de – für Deutschsprechende, politisch korrekt dank umfangreichem Regelwerk
- Futurology – technology and conspiracy theories that shape our future
- hessich_iel – „Schwenk dei Maul mit Ebbelwei, dann komme ka Bazille nei!"
- mildlyinteresting – photos that may have long-term effects on cognitive function
- Space – news from the universe (with nice images of Jupiter and its moons)

Man wählt ein subreddit, zum Beispiel Wissenschaft, mit folgendem Befehl

```
tuir -s Wissenschaft
```

Im Terminal öffnet die Seite schnell und man kann (zum Beispiel mit dem browser `w3m`, siehe Seite 209) auch Bilder anschauen.

6.2.4 HTML in PDF wandeln und dann anzeigen

Viele Webseiten und Textdateien, die mit einem Browser dargestellt werden sollen (Dateiendung `.html` oder `.htm`), sind in der Hypertext-Auszeichnungssprache HTML (HyperText Markup Language) geschrieben. HTML enthält Elemente zur Strukturierung und Formatierung von Text, sowie zum Einfügen von Bildern, Verweisen und so weiter. Elemente stehen in der Regel zwischen einem Paar von sogenannte Tags, zum Beispiel steht der Titel zwischen <title> und </ title>. Es gibt aber auch einzeln stehende Tags, zum Beispiel
. Tags beginnen mit < und enden mit > ; sie werden groß oder klein geschrieben. Konsequente Kleinschreibung ist besser, da die neuere HTML-Variante, XHTML, Kleinschreibung verlangt. HTML-Code steht zwischen den Tags <html> und </html>. Er besteht aus einem Kopfteil zwischen <head> und </head> , und einem Körperteil zwischen <body> und </body>. Tags können Attribute mit oder ohne Wertzuweisung enthalten.

Eine schöne Darstellung vieler Webseiten kann man erreichen, indem der HTML-Code zunächst mithilfe des Programms `wkhtmltopdf` heruntergeladen und im PDF-Format auf unserem Raspberry Pi gespeichert wird. Danach kann man die lokale Datei mit dem Programm `fbgs` (siehe Seite 179) in Ruhe offline betrachten. Die Adresse der Webseite muss man dafür allerdings schon kennen. Das Verfahren wird am Beispiel einer Wikipedia-Seite über das Auge mit Bildern dargestellt.

Wenn das Paket `wkhtmltopdf` installiert wurde, kann man die Webseite mit

```
wkhtmltopdf https://de:wikipedia.org/wiki/Auge auge.pdf
```

herunterladen und in der Datei `auge.pdf` speichern. Dann rufen wir das Programm `fbgs` mit hoher Auflösung auf (hier: 300 dpi), um die Datei `auge.pdf` anzusehen.

```
fbgs -r 300 auge.pdf
```

Mit einem leistungsschwachen Rechner muss man etwas warten, bis die Darstellung erfolgt, aber das Ergebnis ist gut.

6.2.5 E-Mails mit ssmtp verschicken

In manchen Anwendungen möchte man eine E-mail automatisch versenden. Dafür installiert man die Pakete `mailutils` und `ssmtp`.

Außerdem benötigt man ein E-Mail-Konto mit E-Mail-Adresse bei einem E-Mail-Anbieter (zum Beispiel: zxyu@gmx.de bei GMX oder entsprechend bei Gmail). Der Umweg über einen E-Mail-Anbieter ist notwendig, da E-Mails, welche direkt von einem privaten Rechner gesandt werden, von empfangenden E-Mail-Anbietern oft nicht angenommen werden (um Spamming zu erschweren).

GMX Deutschland

Im E-Mail-Konto muss der Zugriff über POP3 und IMAP erlaubt werden. Dazu öffnet man, nach dem Login, im GMX-Menü unter „Einstellungen“ den Punkt „POP3/IMAP Abruf“ und aktiviert „POP3 und IMAP Zugriff erlauben“.

Dann öffnet man mit einem Editor die Konfigurationsdatei `ssmtp.conf`

```
sudo nano /etc/ssmtp/ssmtp.conf
```

und ersetzt deren Inhalt, im Beispiel für ein E-Mail-Konto bei GMX, durch

```
#
# Config file for sSMTP sendmail
#
root=zxyu@gmx.de
mailhub=mail.gmx.net:587
rewriteDomain=gmx.net
hostname=gmx.net
UseTLS=YES
UseSTARTTLS=YES
AuthUser=zxyu@gmx.de
AuthPass=PASSWORT
FromLineOverride=NO
```

wobei PASSWORT das Passwort des E-Mail-Kontos ist. Nach dem Speichern und Schließen dieser Datei öffnet man entsprechend die Konfigurationsdatei `revaliases`

```
sudo nano /etc/ssmtp/revaliases
```

und ersetzt deren Inhalt, im Beispiel für ein E-Mail-Konto bei GMX, durch

```
# sSMTP aliases
#
# Format: local_account:outgoing_address:mailhub
ahg:zxyu@gmx.de:mail.gmx.net:587
```

wobei ahg der Name des -Benutzers ist, der E-Mails versenden darf. Nach dem Schließen der Datei kann man mit einem Befehl in der Konsole eine E-mail versenden, zum Beispiel

```
echo "Hallo" | mail -s "Test" -A anl.txt alfred.gitter@eah-jena.de
```

wobei `"Hallo"` eine Zeichenkette ist, die den zu sendenden Text enthält. Nach der Option `-s` folgt eine (kurze) Zeichenkette für die Betreffzeile der E-mail (hier: `"Test"`). Nach der Option `-A` folgt der Name einer Datei (mit vollständigem Pfadnamen), hier `anl.txt`, die als Anlage mit der E-Mail versandt werden soll. Am Ende der Zeile steht die E-Mail-Adresse, an die gesendet werden soll (Empfänger, in obigem Beispiel: `alfred.gitter@eah-jena.de`).

Ein Nachteil des beschriebenen Verfahrens ist jedoch, dass das Passwort des E-Mail-Kontos unverschlüsselt in der Konfigurationsdatei ssmtp.conf gespeichert wird.

6.2.6 E-Mail-Programm (email client) Alpine

Alpine ist ein bewährtes und weiterentwickeltes E-Mail-Programm, das auf dem Programm Pine aufbaut (siehe http://alpine.x10host.com/). Man installiert es mit

```
sudo apt install alpine alpine-doc alpine-pico
```

und mit dem Befehl

```
alpine -v
```

erfährt man, welche Version installiert wurde. Man startet das Programm mit dem Befehl

```
alpine
```

Tasten zur Bedienung sind am unteren Bildschirmrand erklärt. Außerdem kann man die Pfeiltasten zur Navigation verwenden.

Nach dem ersten Start verlässt man das Programm gleich wieder und bewirkt so, dass die Konfigurationsdatei `.pinerc` angelegt wird.

Die Konfiguration kann im Programm selbst (unter `SETUP (C) Config:`) oder durch Ändern der Konfigurationsdatei erfolgen.

Eine einfache Konfiguration wird hier am Beispiel eines E-Mail-Kontos (`zxyu@gmx.de`) beim E-Mail-Anbieter GMX (Deutschland) gezeigt. Im E-Mail-Konto muss der Zugriff durch E-Mail-Programme erlaubt werden, wie oben im Abschnitt 6.2.5 auf Seite 219 bereits beschrieben: Man öffnet, nach dem Login, im GMX-Menü unter „Einstellungen“ den Menüpunkt „POP3/IMAP Abruf“ und aktiviert „POP3 und IMAP Zugriff erlauben“. Die Daten zu den POP3 und SMTP Servern von GMX (Deutschland) findet man auf einer Hilfe-Seite von GMX.

Zur Konfiguration von Alpine öffnen wir die versteckte Datei `.pinerc` mit einem Texteditor (`nano .pinerc`) und ergänzen einige Zeilen:

```
personal-name=ZeYun Gitter
user-domain=gmx.de
smtp-server=mail.gmx.net:587/tls/user=zxyu@gmx.de
inbox-path={pop.gmx.net/ssl/pop3/user=zxyu@gmx.de}
read-message-folder=~/mail/already-read
```

```
trash-folder=~/mail/trash
feature-list=quit-without-confirm
customized-hdrs=From: zxyu@gmx.de
```

Natürlich muss man die Beispieldaten durch eigene Daten ersetzen.

Das „feature“ `quit-without-confirm` bewirkt, dass beim Verlassen von Alpine (`QUIT`) die Rückfrage, ob man Alpine wirklich beenden will, entfällt.

Die vielen anderen Zeilen belassen wir unverändert. Man kann später in der Konfigurationsdatei oder im Programm unter `SETUP (C) Config:` weitere Änderungen vornehmen, um Alpine an die eigenen Bedürfnisse anzupassen.

Da Alpine in der Standardkonfiguration keine Passwörter speichert, muss man das Passwort für den E-Mail-Anbieter jedes Mal nach dem Programmstart neu eingeben. Es gibt ein einfach zu benutzendes Adressbuch (im Menü: ADDRESS BOOK). Zum Schreiben von E-Mails (im Menü: COMPOSE MESSAGE) hat Alpine eine Version des kleinen Texteditors pico (PIne COmposer), der auch außerhalb von Alpine benutzt werden kann, indem man ihn mit dem Befehl `pico` aufruft.

Mehr Information zu Alpine erhält man mit dem Befehl

```
man alpine | less
```

oder in den Dokumentationsdateien im Verzeichnis `/usr/share/doc/alpine-doc` .

7 Dienst- und Büroprogramme

7.1 Dienstprogramme

7.1.1 Dateimanager (tree, mc, nnn)

Wer mit der Kommandozeile arbeitet, braucht eigentlich keinen Dateimanager. Wenn man sich jedoch daran gewöhnt, kann er die Arbeit durchaus erleichtern.

Verzeichnisbaum mit tree

`tree` ist kein Dateimanager, sondern dient zur Darstellung der Verzeichnisstruktur in Form eines Baumgraphen (und ist insofern verwandt mi Dateimanagern). Nach Installation des Pakets `tree` wird der Befehl `tree` zur Verfügung gestellt. Ohne Argument zeigt er das Arbeitsverzeichnis, aber man kann das Verzeichnis, welches gezeigt werden soll, auch als Argument übergeben. Einige Optionen sind

-a zeige auch versteckte Dateien an
-d zeige nur Verzeichnisse, aber keine Dateien an
-h zeige auch die Dateigröße (in kB, MB oder GB) an
-L 2 maximale Weglänge der Knoten des Baums, hier: 2

Bei große Verzeichnisstrukturen sollte man die Ausgabe an `less` weiterleiten. Beispiel:

```
tree -h ./unterverz | less
```

gibt die Verzeichnisstruktur des Unterverzeichnisses `unterverz` des Arbeitsverzeichnisses mit Angabe der Dateigrößen aus. Zum Verlassen von `less` tippt man `q` ein.

Dateimanager mc

Der *Midnight Commander* `mc` (installiert mit den Paketen `mc` und `mc-data`) ist ein guter Dateimanager. Er zeigt links und rechts je eine Verzeichnisliste, um zum Beispiel das Verschieben einer Datei zu vereinfachen. Zwischen beiden Listen wechselt man mit der Tabulatortaste. Am unteren Rand sind nummerierte Menüpunkte aufgeführt, die man mit der entsprechenden Funktionstaste aufruft.

Wenn man ihn (mit einiger Mühe) richtig konfiguriert, macht er alles, was die Dateimanager von grafischen Benutzeroberflächen können. Und wenn man das Paket `gpm` installiert hat (siehe Seite 48), kann man ihn auch mit einer Maus bedienen. Aber wer will das schon. Eine Anleitung (auf Englisch) findet man auf der Webseite
http://linuxcommand.org/lc3_adv_mc.php .

Dateimanager nnn

Der open-source Dateimanager `nnn` wird zwar im Terminal ausgeführt, wurde aber für Systeme mit der graphischen Benutzeroberfläche X geschaffen. Er wird durch den Befehl

```
nnn
```

aufgerufen. Nachteilig sind spezielle Tastenbefehle, die zur Steuerung dienen, und die ungewöhnliche Art der Konfiguration, die auf Linux-Betriebssysteme mit X ausgerichtet ist (insbesondere die Assoziation von Dateien mit Anwendungsprogrammen durch Hilfsprogramme des X-Systems). Vorteile sind die schnelle Ausführung, Dateisuche mit regulären Ausdrücken und die einfache, übersichtliche Darstellung im Terminal. Das Programm wird weiterentwickelt und bietet Erweiterungsmöglichkeiten. Leider bietet die Paketverwaltung des Raspberry Pi nur eine alte Version von `nnn` an (2.2). Damit funktioniert die Auswahl (selection) ohne X nicht. Also ist auch kein einfaches Verschieben und Kopieren möglich. Auf der Webseite
https://github.com/jarun/nnn/wiki/Developer-guides#compile-for-pi
wird jedoch erklärt, wie man eine neue Version installieren kann. Wir gehen im Folgenden davon aus, dass Version 3.3 erfolgreich installiert wurde.

Nach dem Öffnen mit oben genanntem Befehl wird das aktuelle Arbeitsverzeichnis gezeigt. Zunächst (in blau) Unterverzeichnisse, danach die Dateien.

Man kann zwischen verschiedenen Verzeichnis-Seiten umschalten; sie heißen contexts. Es wird normalerweise nur ein context angezeigt (von vier möglichen). Der aktuelle context wird ganz oben genannt. Zwischen den contexts kann man mit der Tabulatortaste wechseln. Es ist möglich, `nnn` so einzurichten, dass zwei contexts nebeneinander gezeigt werden, aber das ist eigentlich nicht im Sinne des Erfinders.

In der normalen Darstellung ist eine Zeile („aktive Zeile“) durch eine besondere Hintergrundfarbe hervorgehoben. Sie zeigt das Verzeichnis oder die Datei an, die durch weitere Befehle geöffnet oder bearbeitet werden soll.

Einige wichtige Steuerungstasten sind

? ruft eine Hilfsseite für die Steuerungstasten auf, die mit `q` beendet wird
d Kurzform (Dateiname) oder Langform (Änderungsdatum, Größe, Name der Datei)
. zeige versteckte Dateien (Name beginnt mit einem Punkt) an / aus (nicht zeigen)
t öffnet Auswahlliste zur Ordnung der Dateien (Standard: alphabetische Ordnung)
↑ ↓ macht die Zeile oberhalb oder unterhalb zur „aktiven Zeile“
→ öffnet ein Unterverzeichnis
← wechselt in das übergeordnete Verzeichnis (parent directory)
∼ wechselt in das Heimatverzeichnis (hier: /home/ahg)
e öffnet die Datei der aktiven Zeile mit dem Standard-Editor (siehe Seite 51)
Leertaste fügt zur Auswahl hinzu (selection) oder entfernt aus der Auswahl
p kopiert die ausgewählten Dateien hierher (aktive Zeile im Zielverzeichnis)
v verschiebt die ausgewählten Dateien hierher (aktive Zeile im Zielverzeichnis)
n zur Erzeugung einer neuen Datei (`f`) oder eines neuen Verzeichnisses (`d`)
Tabulator wechselt in einen anderen context

q verlässt den aktuellen context (oder, vom Anfangs-context aus, beendet `nnn`)
/*re* verwende Suchfilter *re*, Beispiel: `/.py` zeigt Dateien mit der Endung `.py`
Ctrl-O öffnet eine Datei mit einem externen Programm (wird abgefragt)
Ctrl-R zur Umbenennung einer Datei oder eines Verzeichnisses
x zur Löschung einer Datei oder eines Verzeichnisses (mit `y` zur Bestätigung)
Q beendet `nnn`

Wird `nnn` mit der Option `-S` aufgerufen, berechnet er den Speicherplatz für alle Verzeichnisse und zeigt diesen neben den Verzeichnissen an.

7.1.2 Kommandozeilenrechner bc

Nach Installation des Pakets `bc` kann man das Programm `bc` (basic calculator) mit der mathematischen Standardbibliothek und größerer Genauigkeit (Option `-l` für **l**ibrary) und ohne Begrüßungstext (Option `-q` für **q**uiet) aufrufen.

```
bc -lq
```

Man gelangt in den interaktiven Modus, aber es wird kein Prompt gezeigt. Mit `Ctrl-D` (gleichzeitiges Drücken der Control-Taste und der Taste `d`) beendet man `bc`. Eine englische Anleitung kann man mithilfe eines Browsers, zum Beispiel `w3m` (siehe Seite 209) lesen, indem man den Befehl

```
w3m /usr/share/doc/bc/bc.html
```

eingibt. Mit `q` kann man den Browser `w3m` verlassen.

Man kann mit `bc` kleine Rechenprogramme schreiben, doch wir besprechen hier nur einfache Rechenbefehle. Das Dezimaltrennzeichen ist der Punkt. Neben den üblichen arithmetischen Operatoren `+`, `-`, `*`, `/` und `^` (Exponentialfunktion) sind folgende Funktionen definiert.

sqrt(x) $\sqrt{x}$, wobei x eine Zahl ist
s(x) $\sin(x)$, wobei die Zahl x ein Winkel im Bogenmaß (Einheit rad) ist
c(x) $\cos(x)$, wobei die Zahl x ein Winkel im Bogenmaß (Einheit rad) ist
a(x) $\arctan(x)$, wobei der Rückgabewert ein Winkel im Bogenmaß (Einheit rad) ist
l(x) $\ln(x)$, der natürliche Logarithmus (Basis `e`) der Zahl x
j(n,x) ergibt den Wert der Besselfunktion der Ordnung n (ganzzahlig) von der Zahl x

Beispiele weiterer Befehle sind

4*a(1) ergibt die Zahl π
l(x)/l(10) $\log_{10}(x)$, der dekadische Logarithmus (Basis 10) der Zahl x
quit `bc` beenden (ebenso wie durch `Ctrl-D`)

Das ist soweit ganz nett, jedoch fehlen einige mathematische Funktionen. Man kann nun beim Aufruf von `bc` eine Datei angeben, die Ergänzungen enthält. Mit

```
wget http://x-bc.sourceforge.net/extensions.bc
```

kann man die Erweiterungsdatei `extensions.bc` von Steffen Brinkmann (Softwarelizenz GPL) herunterladen. Wir speichern sie im Verzeichnis `shell`

```
mv ~/extensions.bc ~/shell/extensions.bc
```

und schauen sie mit dem Befehl

```
less ~/shell/extensions.bc
```

an. Wir sehen, welche Funktionen und Konstanten (`pi` und `e`) definiert werden. Drücken der Taste `q` beendet `less`. Zum Aufruf des erweiterten `bc` gibt man den Befehl

```
bc -lq less ~/shell/extensions.bc
```

ein. Wie man in der Kommandozeile (statt interaktiv) rechnet, zeigt folgendes Beispiel,

```
echo "sin(pi/2)" | bc -lq ~/shell/extensions.bc
```

was `1.00000000000000000000` auf dem Bildschim ausgibt.

Wenn wir dieses Rechenprogramm unter dem Kurznamen (Alias) `r` aufrufen wollen, öffnen wir die Datei `bashrc` mit einem Editor (siehe Seite 50) und schreiben ans Ende der Datei die Zeile

```
alias r="bc -lq ~/shell/extensions.bc"
```

Nach dem Schließen der geänderten Datei und einem Neustart des Raspberry Pi können wir den erweiterten Kommandozeilenrechner mit dem Kurzbefehl

```
r
```

aufrufen und die vielen mathematischen Funktionen genießen.

7.1.3 Bildschirmphoto, QR-Code, Uhr

Bildschirmphoto mit fbgrab

Nach Installation des Pakets `fbcat` kann man mit dem Programm `fbgrab` ein Bildschirmphoto machen und in einer Datei (im Beispiel: `/home/ahg/image/bsph.png`) im Bildformat PNG speichern. Der Befehl ist

```
fbgrab /home/ahg/image/bsph.png
```

Information zur Bilddatei erhält man um Beispiel mit dem Befehl

```
mediainfo /home/ahg/image/bsph.png
```

wenn das Paket `mediainfo` bereits installiert wurde (siehe Seiten 78 und 197).

Um ein Bildschirmphoto zu machen, muss die oben angegebene Befehlszeile eingegeben werden. Will man diese aber nicht auf dem Bildschirm sehen, wechselt man zunächst in die zweite Konsole (tty2), indem man gleichzeitig die Tasten `Alt` und `F2` (Funktionstaste 2) drückt. Man loggt mit Nutzernamen und Passwort ein und gibt danach den Befehl

```
fbgrab -c 1 /home/ahg/image/bsph.png
```

der ein Bildschirmphoto von Konsole 1 macht. Danach wechselt man zurück zur Konsole 1, indem man gleichzeitig die Tasten `Alt` und `F1` (Funktionstaste 1) drückt.

QR-Code mit qrencode

Abbildung 7.1: QR-Code

Kurze Informationen können graphisch als QR-Code dargestellt werden und damit einfach von von einem Mobilfunkgerät mit Kamera aufgenommen werden. „QR Code“ ist ein Warenzeichen der Firma Denso Wave.
Nach Installation der beiden Pakete `libqrencode4` und `qrencode` kann man durch einen Befehl in der Kommandozeile eine Zeichenkette in QR-Code wandeln und diesen speichern (standardmäßig im Bildformat PNG). Beispiel:

```
qrencode -l H -s 16 -o qr.png "Mit dem Wissen wächst der Zweifel."
```

Einige wichtige Optionen sind

-- help listet alle Optionen auf (mit `| less` verwenden, da es eine lange Liste ergibt)
-l H setzt das Error Correction **L**evel auf hoch, oder `-l L` für niedrig (Standard)
-o datei2 speichert den QR-Code in der Datei `datei2`
-r datei1 liest die Zeichenkette aus einer Datei namens `datei1`
-s 10 bestimmt die Größe der Blöcke und damit des QR-Codes (Standard ist 3)
t PNG bestimmt das Ausgabeformat, Standard ist PNG, ASCII erzeugt Textzeilen

Wie man den QR-Code sofort auf dem Bildschirm bringt, zeigt folgendes Beispiel.

```
qrencode -l H -s 16 -o qr.png "Hallo Welt!" && fbi qr.png
```

Große Uhr mit tty-clock

Nach Installation des kleinen Pakets `tty-clock` erzeugt der Befehl

```
tty-clock -c -s
```

eine große digitale Uhr in der Mitte des Terminals. Die Option `-c` zentriert die Uhr im Terminal. Mit der Option `-s` werden auch Sekunden angezeigt. Ein Tastendruck auf `q` beendet das Programm.

Ein anderes Format für das Datum erhält man zum Beispiel (siehe Abbildung 7.2) mit

```
tty-clock -c -f "%A, %d. %B %Y"
```

wobei `-f "..."` die Datumsangabe unter der Uhrzeit formatiert. Allerdings werden die Monatsnamen und Wochentage in Englisch geschrieben.

Abbildung 7.2: `tty-clock`

7.1.4 Sprachausgabe und Übersetzer

Sprachausgabe mit eSpeakNG (offline)

Ein einfaches, aber leistungsfähiges Programm zur Erzeugung und Ausgabe von Sprache (Sprachsynthesizer) ist *eSpeakNG*. Es ist der Nachfolger von *eSpeak*. Die Installation erfolgt durch

```
sudo apt install espeak-ng espeak-ng-data espeak-ng-espeak
```

Das Paket `espeak-ng-espeak` ermöglicht durch entsprechende Verweise, dass `espeak-ng` auch mit dem Namen des Vorgängerprogramms *espeak* aufgerufen werden kann.

`espeak-ng` kann die Aussprache an verschiedene Sprachen anpassen und hat mehrere männliche und weibliche Stimmen zur Auswahl. Es kann eine vorgegebene Zeichenkette sprechen oder aus einer Textdatei vorlesen. Durch

```
espeak -vde+m2 "Guten Tag!"
```

erhält man eine Begrüßung in deutscher Sprache (Option `-vde`) mit einer männlichen Stimme (Zusatz `+m2`, empfohlen). Leider wird manchmal (abhängig von der gewählten Stimme) der Anfang des Textes nicht gesprochen.

Mögliche Sprachen sind unter anderem US-amerikanisches Englisch (Option `-ven-US`), britisches Englisch (Option `-ven`) und Mandarin-Chinesisch (Option `-vcmn`).

Auf der Webseite des Projekts, https://github.com/espeak-ng/espeak-ng, findet man eine ausführliche Beschreibung und mit dem Befehl

```
espeak -h | less
```

erhält man eine kurze Übersicht der Optionen.

Das Paket `mbrola` stellt weitere Stimmen zur Verfügung, die in einigen Sprachen (nicht Englisch) besser klingen. Wenn das Paket in den Installationsquellen des Raspberry Betriebssystems fehlt, kann man es mit den drei Befehlen

```
wget http://steinerdatenbank.de/software/mbrola3.0.1h_armhf.deb
sudo dpkg -i mbrola3.0.1h_armhf.deb
sudo apt install mbrola-de?
```

bekommen (Dank an Günter Kreidl). Ein Beispiel mit `mbrola` und männlicher Stimme

```
espeak -v mb-de4 "Guten Tag!"
```

und ein Beispiel mit `mbrola` und weiblicher Stimme

```
espeak -v mb-de3 "Guten Tag!"
```

Sprachausgabe mit gTTS (online)

Mit Internetverbindung (online) kann man die Pythonbibliothek `gTTS` auch im Terminal nutzen, vorausgesetzt sie wurde installiert (siehe Seite 399).

Folgende Befehlszeile spricht auf deutsch (Option `-l de`) „Guten Tag“ mithilfe einer Programmierschnittstelle (API) von Google Translate und dem VLC zur Sprachausgabe (Schnittstellenmodul `dummy`, siehe Seite 192).

```
gtts-cli -l de -o tmp.mp3 "Gguten Tag!" && cvlc --play-and-exit tmp.mp3
```

In obigem Text `Gguten Tag!` soll das zweite `g` den schwachen Anlaut verstärken.

Übersetzungen mit dict (offline)

Die Übersetzung von oder in eine Fremdsprache ist in der Konsole einfach und schnell möglich, wenn die Zeichen im begrenzten Zeichensatz der Konsole enthalten sind. Andernfalls ist ein Umweg notwendig. Eine Übersetzung kann mithilfe eines lokalen Wörterbuchs geschehen oder mit einem externen Programm über das Internet.

Dict ist ein System, das mithilfe von Wörterbuch-Dateien, die im eigenen Rechner gespeichert werden, eine Übersetzung einzelner Begriffe zwischen verschiedenen europäischen Sprachen bereitstellt. Je ein Wörterbuch für Deutsch-Englisch und Englisch-Deutsch wird mit dem Befehl

```
sudo apt install dict dictd dict-de-en
```

installiert. Eine Übersetzung Deutsch-Englisch des Begriffs Haus wird durch

```
dict -d german-english Haus | less
```

erreicht, wobei die Option `-d german-english` die Verwendung des Wörterbuchs für Deutsch-Englisch bestimmt. Entsprechend bestimmt `-d english-german` das Wörterbuch Englisch-Deutsch. Ohne Verwendung der Option `-d` werden alle Wörterbücher durchsucht. Für Dict gibt es weitere Wörterbücher, aber nicht für alle Sprachen. Es arbeitet offline und die Suche ist daher schnell.

Für Autoren ist ein Thesaurus hilfreich. Das ist ein Wörterbuch, das zu einem gegebenen Wort andere Worte auflistet, welche eine ähnliche Bedeutung haben können (Synonyme). Einen englischen Thesaurus liefert das Paket `dict-moby-thesaurus`. Nach der Installation ergibt der Befehl

```
dict -d moby-thesaurus bed | less
```

eine Liste von Synonymen zum englischen Begriff bed.

Übersetzungen mit translate-shell (online)

Nach Installation des Pakets `translate-shell` kann man über das Internet den Übersetzungsdienst von Google nutzen. Zum Beispiel erhält man mit dem Befehl

```
trans en:de "main gate"
```

die Übersetzung des englischen Begriffs main gate ins Deutsche: Haupteingang, Haupttor. Vor dem Doppelpunkt steht die Sprache, aus der übersetzt wird (Quelle) und danach die Sprache, in die übersetzt wird (Ziel). Wenn man statt eines einzelnen Worte einen zusammengesetzten Begriff (wir in obigem Beispiel) oder einen Satz übersetzen lassen will, verwendet man entweder einfache oder doppelte Anführungszeichen.

Die Option `-b` führt zu einer kürzeren Ausgabe. Das kurze Shellscript

```
#!/usr/bin/env bash
e="main gate"
d=$(trans -b en:de "$e")
echo $d
```

gibt das Wort `Haupteingang` aus.

Man kann den zu übersetzenden Text auch aus einer Textdatei lesen (im folgenden Beispiel `inp.txt`) oder das Ergebnis in eine Textdatei schreiben (im folgenden Beispiel `out.txt`) oder beides. Beispiel:

```
trans en:de -i inp.txt -o out.txt
```

Bei einigen Sprachen, wie Englisch und Deutsch gibt es ein erläuterndes Wörterbuch. Dafür setzt man für Quelle und Ziel die gleiche Sprache ein. Zum Beispiel ergibt

```
trans de:de Hypothek
```

eine deutsche Erklärung für den deutschen Begriff Hypothek.

Für Sprachen, deren Schrift mit Unicode-Zeichen ausgegeben werden muss (zum Beispiel Chinesisch), kann man `fbterm` verwenden (siehe Seite 86). Der Befehl

```
trans de:zh "Pferd"
```

liefert im `fbterm` das chinesische Schriftzeichen für „Pferd“.

Man kann das Übersetzte auch aussprechen lassen. Ob der Satz von Erich Kästner

```
trans -b -p de:ja "Es gibt nichts Gutes, außer man tut es."
```

für Japaner Sinn ergibt, weiß ich allerdings nicht.

7.1.5 Geteilter Bildschirm mit tmux

Mitunter möchte man verschiedene Anwendungen nebeneinander auf dem Bildschirm darstellen. Zum Beispiel könnte man auf der linken Bildschirmseite in einem Editor einen englischen Text schreiben, während auf der rechten Bildschirmseite ein Übersetzungsprogramm benutzt wird (siehe Abschnitt 7.1.4). Das ist auch ohne X mit einem *terminal multiplexer*, zum Beispiel `tmux` möglich. Das Paket `tmux` wird mit

```
sudo apt install tmux
```

installiert und mit

```
tmux
```

gestartet. Man erhält eine „Fensterscheibe“ (pane) mit einer Statusleiste. Gesteuert wird `tmux` mit besonderen Tastenbefehlen, die mit `Ctrl-B` beginnen (gleichzeitiges Drücken der Ctrl-Taste und `B`), gefolgt von einer einzelnen Taste. Einige wichtige Tastenbefehle:

Ctrl-B " teilt die aktuelle Fensterscheibe horizontal (Divided pane is double pane.)
Ctrl-B % teilt die aktuelle Fensterscheibe vertikal (in eine linke und eine rechte pane)
Ctrl-B X schließt die aktuelle Fensterscheibe (pane), nach Bestätigung mit `y` (für yes)
Ctrl-B → wechselt horizontal zwischen Fensterscheiben (panes), auch mit Ctrl-B ←
Ctrl-B ↓ wechselt vertikal zwischen Fensterscheiben (panes), auch mit Ctrl-B ↑

Zum Beenden von `tmux` gibt man den Befehl

```
tmux kill-server
```

ein. Zu beachten ist, dass in einer Fensterscheibe (pane) von `tmux` viele Anwendungsprogramme, die den Framebuffer benutzen, nicht laufen. Es ist jedoch möglich, im `fbterm` (siehe Seite 86) `tmux` zu benutzen (in dieser Reihenfolge).

Wenn man zwei Textdateien, `f_1` und `f_2`, parallel mit dem Texteditor `nano` bearbeiten will und mit der Maus arbeitet, um Zeichenketten zu kopieren und in der einen oder der anderen Textdatei wieder einzufügen (siehe Seite 48), kann man eine Konfigurationsdatei `~/.tmux.conf` anlegen, mit folgendem Inhalt:

```
# start tmux with this command: tmux attach
new -s my_s                         # create session
neww -n my_w nano f_1               # create window
# executed command in the new window: nano f_1
split-window -h -t my_w nano f_2  # split window
# t = target, left: pane 0 and right: pane 1
# executed command in the new pane (1): nano f_2
select-pane -t 0                    # go to left pane
set -g mouse on                     # mouse support
# mark text: left mouse button, insert: middle button
# switch between panes with ctrl-b -> and ctrl-b <-
# leave tmux with this command: tmux kill-server
```

Zum Aufruf von `tmux` muss man dann den Befehl

tmux attach

verwenden. Mit der Tastenkombination `Ctrl-B` → wechselt man zwischen den Textdateien in den beiden Fensterhälften und mit `tmux kill-server` verlässt man `tmux`.

7.2 Büroprogramme

Eine Sammlung von „Büroprogrammen“ wie LibreOffice oder Kleinweichbüro gibt es für die Linux-Konsole nicht. Für geringere Ansprüche gibt es Ersatzprogramme und für höhere Ansprüche gibt es bessere, aber umfangreiche Pakete, zum Beispiel LaTeX.

7.2.1 Adressbuch, Terminplanung und Aufgabenmanagement

Adressbuch mit abook

Nach Installation des Paktes `abook` steht ein einfaches, leicht zu handhabendes Adressbuch zur Verfügung, welches durch

```
abook
```

geöffnet wird und in deutscher Übersetzung vorliegt. Die vier wichtigsten Tastenbefehle werden in der Kopfzeile genannt. Einige wichtige Tastenbefehle sind

? zeigt eine Liste der verfügbaren Tastenbefehle
a (add) dient zum Hinzufügen eines Eintrags
r (remove) entfernt einen Eintrag (nach Bestätigung mi `j`)
S (großes S, sort) sortiert die Einträge nach Namen alphabetisch
↑ ↓ bewegen die Auswahl des aktuellen Eintrags nach oben oder unten
ENTER öffnet den aktuellen Eintrag, es erscheinen mehrere Reiter (Tabs)
→ ← wählen einen Reiter (Tab) aus und öffnen ihn
1, 2, 3 . . . wählen einen nummerierten Menüpunkt im geöffneten Reiter (Tab)
q (quit) beendet das Adressbuch

Die Daten des Programms liegen im verstecketen Verzeichnis `.abook` des Nutzers. Das Programm kann mit einer Konfigurationsdatei angepasst werden (was aber nicht notwendig ist), bietet Suchmöglichkeiten und kann Adressdaten formatiert importieren oder exportieren. Es arbeitet gut mit den E-Mail-Programmen `mutt` und `neomutt` zusammen.

Persönliche Terminplanung mit calcurse

Das kleine Paket `calcurse` stellt das gleichnamige Programm zur Terminplanung bereit. Es arbeitet schnell und passt sich an die länderspezifische Einstelleung des Betriebssystems an, kann also auch in deutsch verwendet werden. Es wird mit

```
sudo apt install calcurse
```

installiert. Der interaktive Modus wird mit dem Befehl

```
calcurse
```

gestartet. Die übersichtliche Bildschirmfläche wird in drei Rahmen unterteilt (Appointments / Termine, Calendar / Kalender, TODO / Zu erledigen). Mit den Pfeiltasten kann man im Rahmen Calendar / Kalender einen Tag auswählen und mit der Tabulatortaste kann man den Rahmen wechseln. Unten ist eine selbsterklärende Befehlsliste, die als Menü dient. Nach Drücken der Taste `o` werden weitere Befehle gezeigt. Mit `q` gelang man von einem Unter-Menü zum Hauptmenü zurück und kann dort das Programm mit `q` beenden. Das Erscheinungsbild kann im Menü verändert werden.

Im Verzeichnis `/home/ahg/.config/calcurse` legt `calcurse` beim ersten Aufruf einige Dateien an (wobei für `ahg` natürlich der Nutzername zu setzen ist). Die Konfigurationsdatei `/home/ahg/.config/calcurse/conf` kann über das Befehlsmenü des Programms (im interaktiven Modus) geändert werden oder direkt mit einem Texteditor wie `nano` (siehe Seite 131) bearbeitet werden. Um eine überflüssige Nachfrage beim Beenden des Programms auszuschalten, kann man die Zeile

```
appearance.confirmquit=yes
```

in der Konfigurationsdatei wie folgt ändern:

```
appearance.confirmquit=no
```

Für die Eingabe eines Datums ist das Format `mm/dd/yyyy` voreingestellt. Wenn man lieber das Format `dd/mm/yyyy` verwenden will, ändert man

```
format.inputdate=1
```

in

```
format.inputdate=2
```

Die voreingestellte Farben sind dunkelrot, hellrot und weiß. Wenn man die Zeile

```
appearance.theme=red on default
```

in

```
appearance.theme=green on default
```

ändert, erhält man die Farben grün,hellrot und weiß.

Um einen bestimmten Termin (appointment) einzutragen, wählt man im Rahmen Calendar / Kalender einen Tag aus und drückt die Tastenkombination `Ctrl-A`. Danach gibt man die Uhrzeit ein oder drückt einfach ENTER, wenn der Termin für den ganzen Tag gilt. Danach trägt man die Kurzbeschreibung des Termins ein. Für einen Eintrag im Rahmen TODO / Zu erledigen drückt man die Tastenkombination `Ctrl-T`. Die zu erledigenden Aufgaben sind nicht an ein Datum gebunden, aber man ordnet ihnen

eine priority (Grad der Wichtigkeit) zu, als Zahl zwischen 1 (höchste) und 9 (niedrigste Wichtigkeit). Eine mar

Wenn man das oben empfohlene Farbthema (grün,hellrot und weiß) gewählt hat, erscheint bei der Auswahl des Rahmens Appointments / Termine der aktuelle (zu bearbeitende) Eintrag grün, während die anderen Einträge weiß bleiben. Mit den Pfeiltasten gelangt man zu einem anderen Eintrag. Um einen aktuellen Eintrag (zum Beispiel einen jährlichen Geburtstag) für die kommenden Jahre automatisch zu wiederholen, drückt man die Taste `r` und auf die Frage `A (s)imple or (a)dvanced repetition?` (bei einem englischen Menü) kann man `s` und dann `y` drücken (für year) und schließlich nach `Frequency:` die 1 stehen lassen und zweimal ENTER drücken . Falls man ein deutsches Menu hat, sind die Texte entsprechend. Einträge mit automatischer Wiederholung werden von `calcurse` mit einem $*$ markiert.

Kalenderdaten manuell einzutragen, ist mühsam. Oft werden aber Kalenderdaten als Datei im standardisierten Datenformat iCalendar bereitgestellt. Diese Dateien haben meistens die Endung `.ics`. Man kann sie mit dem Befehl

```
calcurse -i datei.ics
```

importieren, wobei `datei.ics` der Pfad zu Kalenderdatei ist. Dateien mit Feiertagen und Schulferien, meistens auch lokale Müllabfuhrtermine, findet man leicht im Internet.

Im nicht-interaktiven Modus, also mit einem Konsolenbefehl, erhält man durch

```
calcurse -d 3
```

die Termine der nächsten drei Tage.

`calcurse` ist leicht zu bedienen und einfach gut. Wer eine englische Anleitung lesen will, kann dies mithilfe eines Browsers, zum Beispiel `w3m` (siehe Seite 209) durch

```
w3m /usr/share/doc/calcurse/manual.html
```

tun. Mit `q` kann man den Browser `w3m` verlassen. Einige deutsche Textdateien zu einzelnen Themen findet man im Verzeichnis `/usr/share/doc/calcurse/de`.

Persönliche Terminplanung mit when

Noch einfacher als `calcurse`, aber ohne graphische Elemente, ist die Terminplanung mit `when`. Nach der Installation in üblicher Weise wird beim ersten Aufruf mit

```
when
```

das Programm eingerichtet. Auf die erste Frage antwortet man `y` und drückt die ENTER-Taste. Auf die zweite Frage trägt man de Namen des Editors ein, der künftig verwendet werden soll: `nano`. Um Termine einzutragen, gibt man

```
when e
```

ein und trägt einmalige Termine nach dem Schema `Jahr Monat Tag, Text` ein, und jährlich wiederkehrende Termine nach dem Schema `* Monat Tag, Text`, zum Beispiel:

```
2029 04 13, Asteroid Apophis zerstört die Erde
* 05 18, Zeyuns Geburtstag
```

Es ist sinnvoll, aber nicht notwendig, die Einträge in der Reihenfolge von Monat und Tag zu ordnen. Nach jedem Aufruf von `when` werden die Termine der kommenden zwei Wochen angezeigt.

Will man nur einen Dreimonatskalender (ohne Termine) sehen, genügt der Befehl

```
when c
```

Im versteckten Verzeichnis `.when` sind die Datei `calendar` mit den Einträgen und die Konfigurationsdatei `preferences`. Mit `when -help` erhält man eine kurze Beschreibung.

Aufgabenmanagement mit taskwarrior

Das Programm `taskwarrior` zum Aufgabenmanagement (task management) wird mit

```
sudo apt install taskwarrior
```

installiert und unter dem Namen `task` aufgerufen. Der Befehl

```
task calendar
```

zeigt einen aktuellen Kalender, in dem die anstehenden Aufgaben hervorgehoben sind.

```
man  task
```

gibt Hilfe zur Benutzung des Programms.

7.2.2 Einfache Textverarbeitung und Wandlung von .docx

Einfache Textverarbeitung ist schon mit einem Editor wie `nano` möglich. Für das Schreiben langer Texte, Textformatierungen (fett, kursiv, usw.) oder leichtes Speichern in verschiedenen Dateiformaten ist `nano` dagegen nicht geeignet. Typographisch beste Manuskripte erstellt man direkt in *Latex*. Allerdings ist das relativ aufwändig. Eine empfehlenswerte Lösung ist die (vorläufige) Formatierung mit *Markdown* und die anschließende Übertragung in eine vorzeigbares und austauschbares Dokumentenformat mit *Pandoc*. Eine weitere Möglichkeit, für geringe Ansprüche, ist das Textverarbeitungsprogramm *wordgrinder*.

Einfach mit Markdown und Pandoc

Wer schreibt, sollte auf den Inhalt achten und die Formatierung hintan stellen. Das gelingt mit einem einfachen Texteditor. Doch braucht man zur Strukturierung noch Einiges, zum Beispiel Überschriften, und eine Textauszeichnung durch kursive oder fette Schrift. Gut geeignet ist die leichte Textauszeichnungssprache *Markdown*, die sowohl im Terminal als auch auf graphischen Benutzeroberflächen zu gebrauchen ist. Wesentliche Vorteile sind

- leicht erlernbar und anwendbar
- auf allen Betriebssystemen verfügbar
- übertragbar in andere Formate

Auch wenn das Textdokument später in einem anderen Format vorliegen soll, empfiehlt es sich, den Text zuerst mit einem Editor in Markdown zu schreiben. Die Umwandlung in ein anderes Format (DOC, DOCX, HTML, ODT, TEX, XML, ...) geschieht danach mithilfe des Programms *Pandoc*.

Mehr dazu findet man in den Artikeln „Formatting Open Science: agilely creating multiple document formats for academic manuscripts with Pandoc Scholar“ und „Akademisches Schreiben mit Markdown und Pandoc Scholar“ von Robert Winkler.

Es gibt zahlreiche moderne Dienste im Internet, die Markdown verwenden (zum Beispiel GitHub und Stack Overflow), allerdings oft mit besonderen Erweiterungen.

Minimalistische Textverarbeitung mit wordgrinder

Das Programm `wordgrinder` wird mit dem gleichnamigen Paket installiert. Es ist die *very light* Alternative zu den bekannten Textverarbeitungsprogrammen auf graphischen Oberflächen, zum Beispiel LibreOffice Writer.

Man kann schnell den Entwurf eines Manuskripts mit sehr einfachen Formatierungen schreiben, aber muss anschließend mit einem anderen Programm nacharbeiten, um eine gute Formatierung zu erreichen. Im Terminal wird man dafür LaTeX oder Groff einsetzen. Vorteile: Die Bedienung ist einfach. Es läuft schnell, auch auf sehr schwachen Rechnern (zum Beispiel einem Raspberry Pi, Modell 1). Ein Export in andere Dateiformate, zum Beispiel HTML, ist möglich. Nachteile: Der Funktionsumfang ist gering. Bilder und Tabellen können nicht eingebunden werden. Die Speicherung in Kleinweich-Dateiformaten ist nicht möglich.

Eine Beschreibung (englisch) mit Bildern findet man auf der Webseite

`https://www.ihaveapc.com/2018/06/wordgrinder-an-old-school-text-editor-for-linux/`

Nach dem Programmstart sieht man zwei waagerechte Linien, die den Schreibbereich begrenzen. Wer sie nicht sehen will, kann sie über das Menü entfernen. Das Menü öffnet man mit der Escape-Taste (`Esc`).

Es gibt verschiedene Absatzformate, die man im Menü unter `Style` (`S`), `Change paragraph style` (`P`) einstellt. Zum Beispiel: Absätze mit gewöhnlichem Text (`P`) und Überschriften verschiedener Ordnung (`H1`, `H2`, `H3`, `H4`). Die einfache Linux-Konsole kann

die verschieden Absatzformate nur schlecht wiedergeben. Daher sollte man im Menü unter `Style` (`S`), `Set margin mode` (`M`) die Option `Show paragraph styles` (`S`) wählen. So wird am linken Rand das Absatzformat abgekürzt genannt. Beim Export, zum Beispiel im HTML-Format, werden die Absatzformatierungen natürlich mitgenommen.

Wichtig ist das Markieren einer Zeichenkette. Dafür bewegt man den Cursor auf das erste Zeichen der zu markierenden Zeichenkette und drückt die Tastenkombination `Ctrl-Leertaste`.[1] Dann bewegt man den Cursor zum Ende der Kette und dabei wird diese schwarz auf weiß statt weiß auf schwarz dargestellt. Erneutes Drücken der Tastenkombination `Ctrl-Leertaste` beendet die Markierung.

Eine alternative Möglichkeit der nach rechts laufenden Markierung sollte eigentlich durch Drücken der Tastenkombination ⇑→ gegeben sein, wobei ich mit ⇑ die Umschalttaste (Shift) meine. Leider funktioniert diese Methode nicht in meinem System. Daher habe ich im Navigations-Menü die Aktion `Selection right` mit der INSERT-Taste, die auf einer deutschen Tastatur die Aufschrift `Einfg` haben kann, belegt. Wie das geht, wird in der Statuszeile des Programms kurz erklärt. Diese Tastenfestlegung gilt jedoch nur für das aktuelle Dokument.

Eine markierte Zeichenkette kann man mit `Ctrl-X` ausschneiden oder mit `Ctrl-C` in die Zwischenablage (Clipboard) kopieren. Mit `Ctrl-V` wird der Inhalt der Zwischenablage an der Position des Cursors eingefügt.

Eine markierte Zeichenkette kann man mit `Ctrl-B` fett setzen (englisch: Bold Type), mit `Ctrl-I` kursiv setzen (englisch: Italic type) oder mit `Ctrl-U` unterstreichen. Mit `Ctrl-O` setzt man einen markierten Text wieder in normaler Schrift. Diese Textauszeichnungen kann man auch über das Menü vornehmen. Ohne Markierung wirken die Auszeichnungsbefehle im folgenden Text so lange, bis sie durch `Ctrl-O` aufgehoben werden. Allerdings hängt die Darstellung des ausgezeichneten Texts im Terminal vom verwendeten Terminaltyp ab und die Linux-Konsole ist diesbezüglich beschränkt.

Im Menü, unter `File` (`F`), kann man Dokumente aus Dateien laden oder speichern. Ein `document set` kann aus mehreren Dokumenten bestehen und wird in einer Datei mit einem besonderen `wordgrinder`-Format gespeichert. Außerdem kann man ein Dokument in einer Datei mit den Formaten ODT (`.odt`), HTML (`.html`), Markdown (`.md`), plain text (`.txt`), LaTeX (`.tex`) und Troff (`.tr`) speichern und laden.

Automatisches Speichern des bearbeiteten Dokuments in bestimmten Zeitintervallen kann im Menü unter `File` (`F`), `Document settings` (`T`), `Autosave` (`A`) eingestellt werden.

Mit der Tastenkombination `CtrL-Q` verlässt man das Programm. Hat man den Text verändert und nicht gespeichert, wird man gefragt, ob der man das Programm tatsächlich ohne Speicherung verlassen will (`Y`) oder doch lieber nicht (`N`).

Wandlung von .docx

Viele unschuldige Menschen arbeiten mit dem Textverarbeitungsprogramm WORD der Firma Microsoft und speichern ihre Machwerke im Dateiformat Office Open XML Do-

[1] Die Ctrl-Taste heißt auf einer deutschen Tastatur oft `Strg`. Die Leertaste ist die lange waagerechte Taste unten auf der Tastatur. Im Englischen wird sie auch *space bar* (Weltraumkneipe) genannt.

cument mit der Dateiendung `.docx`. Den unformatierten Text einer WORD-Datei mit der Endung `.docx` kann man mit einem Pythonprogramm herausholen, siehe Seite 295.

Eine einfache Möglichkeit, eine WORD-Datei `word.docx` in eine anderes Format zu wandeln ist die Verwendung des Textverarbeitungsprogramms *Abiword*, das durch

```
sudo apt install abiword
```

installiert wird. Abiword arbeitet normalerweise mit einer graphischen Oberfläche (X) und die haben wir ja nicht. Wir können aber Abiword in der Kommandozeile veranlassen, die Datei `word.docx` in einem anderen Format zu speichern. Der Befehl

```
abiword --to=txt word.docx
```

erzeugt eine Datei `word.txt` mit dem Textinhalt ohne Auszeichnungen (wie Fettdruck oder Unterstreichung) und ohne eventuell enthaltene Bilder. Die gruseligen Warnungen, die auf dem Terminal erscheinen, beachten wir nicht. Mit dem Befehl

```
abiword --to=rtf word.docx
```

wandeln wir `word.docx` in eine Datei `word.rtf` im Rich Text Format (RTF), ein älteres Format der Firma Microsoft. Und mit

```
abiword --to=pdf word.rtf
```

wandeln wir `word.rtf` in eine PDF-Datei `word.pdf`. Die PDF-Datei können wir mit dem Programm `fbgs` auf unserem Bildschirm graphisch darstellen, siehe Seite 179. Man könnte `word.docx` mit dem Befehl `abiword --to=pdf word.docx` direkt in eine PDF-Datei wandeln, aber das Ergbnis ist oft schlechter (weil die Einrichtung von vertikalem Leerraum schwierig ist).

7.2.3 Tabellenkalkulation und Textpräsentation

Tabellenkalkulation für Einzelgänger mit sc

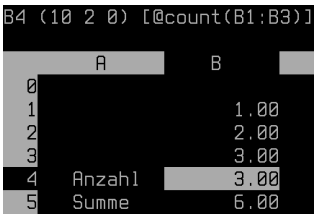


Abbildung 7.3: `sc`

Tabellenkalkulation ohne Maus? Simon's Cat sagt Ja.

Nachdem man das Paket `sc` installiert hat, kann man das gleichnamige Programm zur Tabellenkalkulation aufrufen, siehe Abbildung 5.6. Es beginnt im Befehlsmodus.

Wichtige Befehle sind:

? Hilfe (Menü mit verschieden Themen, Q zum Beenden)
← **oder h** nach links bewegen
→ **oder l** nach rechts bewegen
↓ **oder j** nach unten bewegen
↑ **oder k** nach oben bewegen

x den Inhalt einer Zelle entfernen
\Preis Text-Eingabe in eine Zelle, abgeschlossen mit der ENTER-Taste
Ei Textzelle bearbeiten, Backspace-Taste zum Löschen, Beenden mit ENTER
=8.123 Eingabe einer Zahl, dann ENTER, gezeigt werden zwei Nachkommastellen
=A0+B0*C0 Eingabe einer Formel, die mehrere andere Zelleninhalte verwendet
ei Zahlenzelle oder Formelzelle bearbeiten, Backspace löscht, Beenden mit ENTER
ma Übertrage den Zelleninhalt (text, Zahl, Formel) in den Zwischenspeicher a
ca Kopiere den Inhalt von Zwischenspeicher a (mit relativer Adressierung)
=@sum(B0:B5) Summe der Zahlen eines Zellenbereichs, Text wird nicht beachtet
=@count(B0:B5) Anzahl der Zellen, die Zahlen enthalten, in einem Zellenbereich
=@avg(B0:B5) Mittelwert der Zahlen eines Bereichs, Text wird nicht beachtet
=@stddev(B0:B5) Standardabweichung für eine Stichprobe in einem Bereich
Pdatei Speichern in der Datei `datei` in einem besonderen `sc`-Format
Gdatei Laden einer Datei `datei`, die zuvor mit im `sc`-Format gespeichert wurde
q oder Q (quit) sc beenden

Die genannten Befehle erlauben eine schnelle Anwendung. Aber es gibt viele weitere Möglichkeiten. Zellen können farbig sein. Komplizierte Formeln können in einem externen Programm berechnet werden, das von einer Zelle automatisch aufgerufen wird.

Minimalistische Textpräsentation mit TPP

Das Paket `tpp` stellt das gleichnamige Programm (Text Presentation Program) zur Präsentation von Text zur Verfügung. Vorteile: Man braucht wenig zu lernen und es ist leicht zu bedienen. Nachteile: Die Formatierungsmöglichkeiten sind spärlich und es können keine Bilder eingebunden werden. Verwendbar ist es nur für eine anspruchslose Darstellung in einer Arbeitsgruppensitzung.

Eine TPP-Datei ist eine Textdatei. Sie sollte die Endung `.tpp` haben. Hier ein Beispiel.

```
--bgcolor white
--fgcolor black

--author J. Verne
--title Dunkle Energie aus Lunarzellen
--date today %d. %m. %Y

--newpage
--beginoutput
--center Über den Mond
--endoutput

--color red
--center Der Mond ist mächtig.
--color black
```

```
-- center Er hat viel dunkle Energie.

--newpage
--beginoutput
--center Über die Energie
--endoutput

Bei Vollmond ist die Lunarenergie hell,
bei Neumond ist die Lunarenergie dunkel.
```

Befehle werden am Zeilenanfang gegeben und beginnen mit zwei Minuszeichen. Die ersten beiden Zeilen des Beispiels sorgen für Präsentationsseiten („Dias“) mit schwarzem Text auf weißem Grund. Danach folgen Angaben für die Titelseite. Mit `--newpage` beginnt die nächste Präsentationsseite. Der aus `-beginoutput` und `-endoutput` gebildete Block erzeugt einen Rahmen um den eingeschlossenen Text. Zeilen, die mit `--center` beginnen, werden zentriert. Mit `--color red` wird die Schriftfarbe auf Rot umgestellt und mit `--color black` wieder auf Schwarz.

Zur Steuerung der Präsentation dienen insbesondere folgende Befehle.

Leertaste nächste Präsentationsseite laden

← vorherige Präsentationsseite laden

h Anzeige von Hilfe zu den Tastenbefehlen ein / aus

q (quit) Präsentation beenden

Für eine gute Präsentation in der Konsole erzeugt man mit LaTeX eine PDF-Datei und zeigt diese mit dem Programm `fbgs` an (siehe Seite 179).

7.2.4 LaTeX für Textverarbeitung und Präsentation

LaTeX ermöglicht das Schreiben hervorragend formatierter Texte und Präsentationen auf verschiedenen Betriebssystemen und erzeugt eine überall lesbare Ausgabedatei (in der Regel im PDF-Format). Nachteilig ist am Anfang das aufwändige Erlernen.

Grundlegende Installation

Da LaTeX sehr umfangreich ist, sollte man nicht alle LaTeX-Pakete installieren, sondern nur ausgewählte.

```
sudo apt install texlive texlive-latex-extra texlive-lang-german
```

Für die Erstellung eines Literaturverzeichnisses im Dokument sollte man mit

```
sudo apt install texlive-bibtex-extra biber
```

zwei weitere Pakete installieren, wie bereits oben empfohlen (Seite 79).

Nun erzeugen wir ein Verzeichnis für zusätzliche LaTeX-Pakete, die wir später vielleicht installieren wollen, mit folgenden drei Befehlen

```
mkdir ~/texmf/
mkdir ~/texmf/tex/
mkdir ~/texmf/tex/latex/
```

und erzeugen wir ein Verzeichnis für Dokumentationen zu den zusätzlichen Paketen

```
mkdir ~/texmf/doc/
mkdir ~/texmf/doc/latex/
```

Kurze Einführung

Ein LaTeX-Dokument wird durch Quellcode in einer Textdatei gebildet. Dafür genügt ein einfacher Texteditor wie `nano` (siehe Abschnitt 4.2.1 auf Seite 131), der zum Beispiel mit `nano mybsp.tex` aufgerufen wird. Das Dokument wird in einer Datei mit der Endung `.tex` gespeichert, zum Beispiel `mybsp.tex`. Mit dem Befehl

```
pdflatex mybsp
```

wird daraus ein PDF-Dokument compiliert, das man mit `fbgs -xxl mybsp.tex` anschauen kann (siehe Abschnitt 6.1.3 auf Seite 179). Allerdings kann `fbgs` vielleicht nicht auf die richtige Schriftart (Fonts) zugreifen und verwendet einen Ersatz. Ein Script zur einfachen Compilierung mit Löschung von Hilfsdateien findet man auf Seite 151. Das Ausdrucken des PDF-Dokuments soll OHNE Anpassung der Seitengröße geschehen. Das Druckprogramm muss entsprechend eingestellt werden.

Blöcke heißen in LaTeX Umgebungen. Eine Umgebung X beginnt mit `\begin{X}` und endet mit `\end{X}`. Der Textinhalt steht in der document-Umgebung. Alles vor der document-Umgebung heißt Präambel.

In der Präambel wird der Typ des Dokuments festgelegt und die Schriftgröße (im Beispiel unten: 12pt). Außerdem werden Pakete (packages) mit bestimmten Funktionen geladen und Einstellungen vorgenommen.

Der Textinhalt kann durch Formatierungsanweisungen gestaltet werden. Dabei gibt es keine verborgenen Zeichen. Die Formatierungsanweisungen werden in der Regel durch einen Befehl, der mit einem Backslash \ beginnt, eingeleitet.

Ergänzung: Literaturliste

Die Erstellung einer Literaturliste (Literaturverzeichnis) kann mit dem Paket *biblatex* und dem externen Programm *biber* geschehen. Zunächst braucht man eine Bibliographiedatei. Das ist eine Textdatei, die als Literaturdatenbank dient. Ihr Name sollte die Dateiendung `bib` haben. Da es üblich ist, die Bibliographiedatei mit der Zeit zu erweitern, sollte ihr Name eindeutig die Version benennen. Ein möglicher Name ist

`lit_`*JJMMTT*`.bib`, wobei *JJMMTT* das sechsstellige Datum (Jahr, Monat, Tag) der letzten Änderung darstellt.

Die Bibliographiedatei kann mit einem einfachen Texteditor bearbeitet werden. Später wird mit dem LaTeX-Paket *biblatex* und dem externen Programm *biber* eine Literaturliste aus der Bibliographiedatei entnommen. In der Bibliographiedatei gibt es für jede Literaturquelle einen *Eintrag*, der einen eindeutigen Namen hat (BibTeX-Key). Dieser Name kann aus dem Nachnamen des Erstautors, dem Veröffentlichungsjahr und einem Buchstaben bestehen, wobei Kleinbuchstaben verwendet werden. Der Buchstabe am Ende dient zur Unterscheidung mehrerer Arbeiten mit gleichem Erstautor und Jahr.

Jeder Eintrag beginnt mit @ und dem Dokumenttyp (Referenzart) der Literaturquelle. Es folgen `{`, der BibTeX-Key und Felder. Jedes Feld enthält ein Paar bestehend aus Schlüssel = Wert. Dokumenttyp und Schlüssel können mit Groß- oder Kleinbuchstaben geschrieben werden. Leerzeichen außerhalb der Felder sind unbedeutend und dürfen beliebig gesetzt werden. Der Eintrag wird mit `}` beendet. Das Format einer Bibliographiedatei ersieht man aus folgendem Beispiel, das nur einen Eintrag enthält.

```
@book{hook2013a,
author = {Hook, James and Crook, A.},
title = {Extinction of the crocodiles},
publisher = {Broken Jaw Press},
address = {Fredericton},
date = {1908},
}
```

Die Reihenfolge der Einträge ist beliebig, aber der Übersichtlichkeit halber ist es gut, sie nach dem BibTeX-Key alphabetisch zu ordnen. Man kann mit einer großen Bibliographiedatei arbeiten, die mehr Einträge enthält als benötigt werden, da die Erstellung der Literaturliste weniger Rechenzeit braucht als andere Vorgänge bei der Compillierung.

Beispiel

Folgendes Beispiel[2] zeigt den Quellcode für ein LaTeX-Dokument mit Literaturliste und Index, in dem zusätzlich allgemeine Erklärungen zur Syntax von Latex eingefügt sind.

```
\documentclass[portrait,12pt,BCOR=0mm]{scrreprt}
% European typography -> KOMA-Script documentclasses
% classes are scrbook, scrreprt, scrartcl, scrlettr2
% a4paper is the default for all KOMA-Script classes
% page orientation = portrait (default) or landscape
% default font size = 11pt, for scrlettr2 it is 12pt
% academic papers should use larger font size (12pt)
% KOMA-Script uses package typearea for page layout,
```

[2] Wenn man den Verweis zum Herunterladen benutzt, sollte anschließend die (technisch bedingte) Dateiendung `.txt` in die richtige Endung `.tex` verändert werden.

```
% automatically loaded. DIV = number of page columns
% BCOR in KOMA-Scripts not with the geometry package

% "DIV" defaults for A4 are 10pt:8, 11pt:10, 12pt:12
% for single pages: binding correction (BCOR) is 0mm
% default type area and margins (A4 and BCOR=0), mm:
% 10pt DIV  8, 131.25 x 185.63 top 37.13 inner 26.25
% 11pt DIV 10, 147.00 x 207.90 top 29.70 inner 21.00
% 12pt DIV 12, 157.50 x 222.75 top 24.75 inner 17.50

% with KOMA-Script: use geometry only when necessary
% \usepackage[margin=2cm,includefoot]{geometry}
% documentclass options a4paper,portrait -> geometry
% paper = total body (printable area) + margins
% margin = 2 cm from each edge of the paper
% includefoot includes foot into total body

\usepackage[english]{babel} % american english text
% \usepackage[ngerman]{babel} % german, new orthogr.
\usepackage[utf8]{inputenc} % input UTF-8 characters
\usepackage[T1]{fontenc} % T1 font encoding, not OT1
\usepackage{lmodern} % use the Latin Modern fonts

\usepackage[sortlocale=auto,sorting=nyt, % sorting
style=numeric-comp]{biblatex} % or authoryear-comp
\addbibresource{lit_240112.bib}
\usepackage{csquotes,xpatch}

\usepackage{xcolor}   % package for using colors
\usepackage{graphicx} % package for images, graphics
\usepackage{makeidx}  % standard package for indexes
\makeindex % when compiling, make index file ___.idx

% \pagestyle{empty} % header and footer are empty
\pagestyle{plain} % footer contains page number
% warning: the handling of page numbers is difficult

\begin{document}

%\iffalse % if not commented out: do not compile this
% -----------------------------------------------
\title{Rolling Stones}
\subtitle{A report on petrokinesis}
\author{Sisy Phos}
```

```
\date{\today}
\maketitle % print the title page with above items

\tableofcontents % generate the table of contents
% ------------------------------------------------
%\fi

%\iffalse % if not commented out: do not compile this
% ------------------------------------------------
\chapter{Reports shoud have chapters}

\section{Title of first section of first chapter}
Something \emph{important} and a \textbf{bold} word.

% a floating environment (float) must be on one page
% floats (figures, tables): positioned automatically
\begin{figure}[h] % place image (approximately) here
\centering % horizontal image position in the middle
\def\fig{~/image/erdbev.jpg} % path to image file
\IfFileExists{\fig} % check, if image file exists
{ \includegraphics[scale=0.3]{\fig} }
{ \textcolor{red}{Warning: missing image file} }
\caption{This is a figure caption below the image.}
\end{figure}

Note: the first line of a new paragraph is indented.
The word tetrapod\index{tetrapod} will be found in
the index. Whitespace between
        words           is reduced to one space.
% key word in {} -> index, with actual page number
% ------------------------------------------------
%\fi

% \iffalse % if not commented out: do not compile this
% ------------------------------------------------
\chapter{Here comes another chapter}

\section{Title of first section of second chapter}

\subsection{Crime doesn't pay}

\subsubsection{Piracy}
Pirates don't have it easy! \cite{hook2013a}.
% ------------------------------------------------
```

```
% \fi

%\iffalse % if not commented out: do not compile this
% -------------------------------------------------
\printbibliography % print the bibliography here
\printindex % print the index here (using ---.idn)
% -------------------------------------------------
%\fi

\end{document}
```

Das Beispiel kann als Vorlage für eigene Dokumente dienen. Es soll wesentliche Teile und wichtige Befehle zusammenfassen, aber kann eine detaillierte Erläuterung von LaTeX natürlich nicht ersetzen.

Die Zeile `\usepackage[english]babel` passt Latex an Texte in US-amerikanischem Englisch an, während stattdessen `\usepackage[ngerman]babel` für deutsche Texte verwendet wird. Damit werden auch automatisch erzeugte Namen an die jeweilige Sprache angepasst (englisch: figure, deutsch: Abbildung).

Die KOMA-Script Dokumentklassen verwenden automatisch das Paket `typearea` und versuchen damit ein typographisch gutes Seitenformat (page layout) festzulegen. Wenn es aber Vorgaben gibt, welche andere Seitenränder erfordern, kann man diesen mit dem Paket `geometry` selbst festlegen. Die Zeile `\pagestyle{empty}` bewirkt, dass keine Seitenzahlen gesetzt werden.

Mit dem (nicht gedruckten) Prozentzeichen % wird in LaTeX ein Kommentar (bis zum Zeilenende) eingeleitet. Für ein gedrucktes Prozentzeichen muss man daher `\%` schreiben. Die Kommentare in obigem Beispiel dienen hier nur als allgemeine Erklärungen und man wird sie in eigenen Texten weglassen.

Mit der Formatierungsanweisung `\emph{...}` wird der Text zwischen den geschweiften Klammern betont (emphasized), das heißt (in der Regel) kursiv gedruckt. Entsprechend wird der Text durch `\textbf{...}` fett gedruckt (boldface).

Die Compilierung eines LaTeX-Dokuments vom Quellcode zur PDF-Datei ist meistens ein mehrstufiger Prozess. Wenn ein Inhaltsverzeichnis, eine Literaturliste und ein Index erstellt werden müssen, wie in obigem Beispiel (Datei `mybsp.tex`) braucht man nacheinander folgende sechs Befehle:

```
pdflatex mybsp
biber mybsp
pdflatex mybsp
pdflatex mybsp
makeindex mybsp
pdflatex mybsp
```

Dabei werden die Dateiendungen weggelassen. Sie werden automatisch ergänzt. Natürlich kann man wiederholte Befehle aus der History der bash abrufen (siehe Seite 110).

Um die Fehlersuche zu vereinfachen und die Compilierungszeit gering zuhalten, ist ein strukturiertes Vorgehen bei der Erstellung des Quellcodes zu empfehlen. Man sollte von einer geeigneten Vorlage ausgehen, zum Beispiel der obigen. Zunächst sollte man den Textinhalt mit minimaler Formatierung schreiben und erst danach die Formatieung vornehmen. Diese Arbeiten sollten getrennt für Titelseite, einzelne Kapitel, sowie Literaturliste und Index geschehen. Durch Einfassen in einen Block aus `\iffalse` und `\fi` wird ein Abschnitt des Quellcodes der document-Umgebung für den Compiler unsichtbar. Man kann so ein einzelnes Kapitel schnell compilieren und die gefundenen Fehler eingrenzen. Wenn alles fehlerlos läuft, werden die Zeilen mit `\iffalse` und `\fi` auskommentiert, also durch Voranstellen eines %-Zeichens unwirksam gemacht. Graphiken, die mit TikZ erstellt werden (siehe Seite 247), können die Dauer der Compilierung verlängern und sollten erst am Ende eingebunden werden. Man beachte, dass Pakete die mithilfe von TikZ arbeiten, auch oft langsam sind.

Will man den Quellcode auf syntaktische Fehler prüfen, braucht man nicht die obige lange Befehlssequenz, sondern kann eine verkürzte Compilierung vornehmen, welche jeweils den ersten Fehler zeigt. Man kann ein Shellscript[3] dafür schreiben

```
echo -e "\npdflatex auf Fehlersuche:"
pdflatex -draftmode -interaction=batchmode \
mybsp 2>&1 > /dev/null
if [ $? = 0 ] ; then
echo -e "\nOhne Fehler!"
cat mybsp.log | grep "Warning"
else
echo -e "\nFehler! (l. = line number)"
echo -e "\n"
cat mybsp.log | grep -m 1 -A 2 "^!"
echo -e "\n"
fi
```

und zum Beispiel in einer Datei namens `try` im Verzeichnis `~/shell/` speichern. Das Shellscript wird durch `chmod a+x ~/shell/try` ausführbar gemacht und kann dann mit dem Befehl `try` aufgerufen werden. Es zeigt Fehler an und, falls keine Fehler da sind, Warnungen. Nach der Korrektur aufgrund einer Warnung sind oft zwei Läufe von `try` notwendig, damit die Warnung verschwindet.

Installation zusätzlicher Pakete

Das installierte LaTeX-System genügt für viele einfache Texte. Manchmal braucht man aber zusätzliche Pakete. Man kann sie von der Webseite `https://www.ctan.org/` des CTAN (Comprehensive TeX Archive Network) herunterladen. Meistens werden sie komprimiert im ZIP-Dateiformat (Endung `.zip`) angeboten.

[3] Wenn man den Verweis zum Herunterladen benutzt, sollte anschließend die (technisch bedingte) Dateiendung `.txt` entfernt werden.

Allerdings kann es bei der Anwendung neuer Pakete zu Fehlermeldungen kommen, weil das veraltete LaTeX-System, welches über die Paketverwaltung installiert wird, nicht immer richtig mit neuen LaTeX-Paketen zusammenarbeitet.

Wir zeigen die Installation am Beispiel des Pakets `ulem`, welches in der package documentation beschrieben wird, die man in der Webseite https://www.ctan.org/pkg/ulem findet. Zunächst gehen wir in das Verzeichnis ~/latex/ (siehe Seite 48)

```
cd ~/latex/
```

Dann holen wir die komprimierte Paketdatei von der CTAN

```
wget https://mirrors.ctan.org/macros/latex/contrib/ulem.zip
```

Wir erhalten die Datei `ulem.zip` und entpacken sie mit dem Befehl

```
unzip ulem.zip
```

Mit dem Befehl

```
ls
```

vergewissern wir uns, dass wir ein neues Verzeichnis `ulem` haben. Wir löschen die komprimierte Datei `ulem.zip` mit dem Befehl

```
rm -r ulem.zip
```

gehen in das Verzeichnis `ulem`

```
cd ulem
```

sehen uns den Inhalt des Verzeichnisses an

```
ls
```

und erkennen, dass es eine Datei mit der Endung `.sty` gibt. Wir verschieben sie in das Verzeichnis ~/texmf/tex/latex/ mit dem Befehl

```
mv ulem.sty ~/texmf/tex/latex/
```

Man beachte den Schrägstrich / am Ende von ~/texmf/tex/latex/ ! Er stellt klar, dass wir in ein Verzeichnis verschieben und nicht einen neuen Dateinamen geben.

In der Datei `ulem.pdf` wird das Paket beschrieben. Wir erzeugen ein Verzeichnis für die Dokumentation

```
mkdir ~/texmf/doc/latex/ulem/
```

und verschieben die Dokumentation `ulem.pdf` in das neue Verzeichnis, um später darin lesen zu können.

```
mv ulem.pdf ~/texmf/doc/latex/ulem/
```

Schließlich gehen wir in das Verzeichnis `/home/ahg/latex/` und räumen auf.

```
cd ~/latex/
rm -r ulem/
```

Nun sind wir fertig mit der Installation des Pakets `ulem` und können es benutzen.

Bei vielen Paketen erhalten wir nach dem Entpacken keine Datei mit der Endung `.sty`, sondern nur andere Dateien, insbesondere zwei mit dem Endungen `.dtx` und `.ins`. Wenn `x` der Paketname ist, erzeugt man mit dem Befehl

```
latex x.ins
```

neue Dateien. Auf dem Bildschirm wird mitgeteilt, welche Dateien wichtig sind; darunter ist `x.sty`. Nun verfährt man weiter, wie oben beschrieben. Das heißt, man verschiebt die wichtigen Dateien (mindestens `x.sty`) in das Verzeichnis `~/texmf/tex/latex/` und die Dokumentation `x.pdf` in das neue Verzeichnis `~/texmf/doc/latex/x/`.

Zeichnungen und Diagramme mit TikZ

Wenn die LaTeX-Pakete `pgf` (TikZ) und `pgfplots` installiert wurden, kann man Zeichnungen mathematischer Funktionen direkt im LaTeX-Dokument berechnen und einbinden. Folgendes Beispiel stellt das Lennard-Jones-Potential dar (siehe Abbildung 7.4)

```
\documentclass{scrartcl}
\KOMAoptions{fontsize=11pt,} % Schriftgröße
\usepackage[ngerman]{babel} % verwende deutsche Sprache (Umlaute)
\usepackage{tikz} % Tikz
\usepackage{pgfplots} % Plots zeichnen in TeX
 \pgfplotsset{compat=1.13} % Code kompatibel mit Version 1.13
\usepackage{nicefrac} % für schöne a/b mit \nicefrac{nom}{denom}
\begin{document}
 \begin{tikzpicture}
  % relatives Lennard-Jones-Potential: y = 4 ( x^-12 - x^-6 )
  % wobei x = r / sigma und V(r) = 0 => r = sigma
  % und y = V / epsilon und V(r) = Minimum => V = epsilon
   \begin{axis}%
   [width=\linewidth,height=8cm,%ticks=none,
   axis x line=middle,thick,xmin=0.86,xmax=2.05,
   axis y line=left,
   ymin=-1.2, ymax=3.5,
   xlabel={\Large $\nicefrac{r}{\sigma}$},
   % ylabel={$y$},
   legend style={draw=none, fill=none, row sep=1.5mm},
   ]
\addplot[very thick,red,domain=0.82:2.0,samples=200]{4*(x^-12)};
```

```
\addlegendentry{$4\,\left(\frac{\vphantom{X_{X_x}}\sigma}
{r}\right)^{12}$}
\addplot[very thick,green,domain=0.82:2.0,samples=200]{4*(-x^-6)};
\addlegendentry[text depth=3mm]{$-\,4\,\left(
\frac{\vphantom{X_{X_x}}\sigma}{r}\right)^{6}$}
\addplot[very thick,blue,domain=0.82:2.0,samples=200]
{4*(x^-12-x^-6)};
\addlegendentry{\Large $~\frac{\vphantom{X_{X_X}}V_{\mathrm{LJ}}}
{\epsilon} = 4\,\left(\,\left(\frac{\vphantom{X_{X_x}}\sigma}
{r}\right)^{12} - \left(\frac{\vphantom{X_{X_x}}\sigma}
{r}\right)^{6}\,\right)$}
   \end{axis}
 \end{tikzpicture}
\end{document}
```

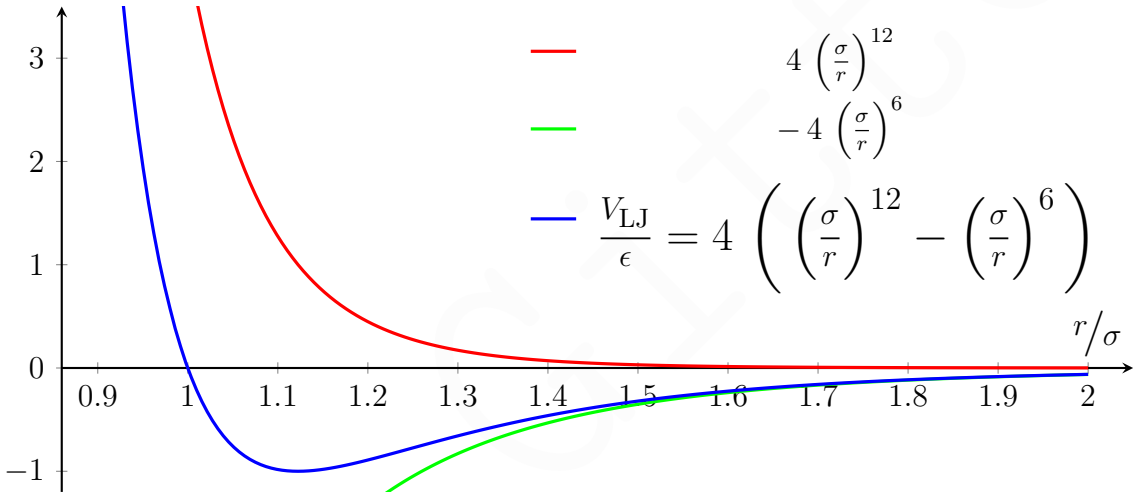


Abbildung 7.4: Zeichnung von Funktionsgraphen mit LaTeX (TikZ und pgfplots)

Ebenso kann man genaue wissenschaftliche Zeichnungen erstellen. Folgendes Beispiel berechnet und zeichnet eine asymmetrische Sammellinse (siehe Abbildung 7.5).

```
\documentclass[a4paper,12pt]{scrartcl}
    \usepackage[ngerman]{babel}
    \usepackage[T1]{fontenc}
    \usepackage{amsmath}
    \usepackage{tikz}
    \usetikzlibrary{arrows.meta}
\begin{document}
\begin{tikzpicture}[scale=1.48]
% Folgende Werte können verändert werden (Eingaben):
  \pgfmathsetmacro{\LX}{14}      % x-Wert für die Linse
```

```
  \pgfmathsetmacro{\BI}{1.6}     % Brechungsindex der Linse
  \pgfmathsetmacro{\LRL}{12}     % linker Krümmungsradius der Linse
  \pgfmathsetmacro{\LRR}{4}      % rechter Krümmungsradius der Linse
  \pgfmathsetmacro{\LD}{4}       % Durchmesser der Linse in cm
  \pgfmathsetmacro{\BW}{5}       % Brennweite der Linse in cm
  \pgfmathsetmacro{\MS}{1/2.5}   % Massstabsfaktor für obige Werte
%
% Berechnung verschiedener Größen aus den Eingaben:
  \pgfmathsetmacro{\YH}{\MS*\LD/2}  % y-Wert für den Bogen-Anfangspunkt
  \pgfmathsetmacro{\KLRL}{\MS*\LRL} % korr. li. Linsenkrümmungsradius
  \pgfmathsetmacro{\KLRR}{\MS*\LRR} % korr. re. Linsenkrümmungsradius
  \pgfmathsetmacro{\WIL}{asin(\LD/(2*\LRL))} % 1/2 Öffnungswinkel links
  \pgfmathsetmacro{\WIR}{asin(\LD/(2*\LRR))} % 1/2 Öffnungswinkel rechts
  \pgfmathsetmacro{\KX}{\MS*\LX} % korrigierter x-Wert für die Linse
  \pgfmathsetmacro{\TXR}{\KX-(\KLRR/2)} % x-Wert für Beschriftung R2
  \pgfmathsetmacro{\TXL}{\KX+(\KLRL/2)} % x-Wert für Beschriftung R1
  \pgfmathsetmacro{\BPL}{\KX-\MS*\BW}   % x-Wert für linken Brennpunkt
  \pgfmathsetmacro{\BPR}{\KX+\MS*\BW}   % x-Wert für rechten Brennpunkt
%
\draw[blue,dotted] (0,0)--(1.88*\KX,0);
\draw[green] (\KX-\KLRR,0)--(\KX,\YH);
\draw[red] (\KX,\YH)--(\KX+\KLRL,0);
\node at (\TXR,\YH) {$\textcolor{green}{\lvert r_2\rvert}$};
\node at (\TXL,\YH) {$\textcolor{red}{\lvert r_1\rvert}$};
\fill[blue] (\BPL,0) circle (0.6mm) ;
\node at (\BPL,-0.4) {$\textcolor{blue}{F}$} ;
\fill[blue] (\BPR,0) circle (0.6mm) ;
\node at (\BPR,-0.4) {$\textcolor{blue}{F^{\,\prime}}$} ;
%
\node at (0.6,0.7) {\small \textcolor{yellow!80!black}{Licht}} ;
\draw[-Stealth,yellow!80!black,very thick] (0,0.3)--(1.5,0.3);
\draw[-Stealth,yellow!80!black,very thick] (0,-0.15)--(1.5,-0.15);
\draw[-Stealth,yellow!80!black,very thick] (0,-0.6)--(1.5,-0.6);
\node at (2.3,0.2) {\footnotesize \textcolor{blue}{optische}} ;
\node at (2.3,-0.25) {\footnotesize \textcolor{blue}{Achse}} ;
%
% Konvex-konvexe Linse
\draw [fill=blue!20] (\KX,\YH)
arc[start angle=180-\WIL,delta angle=2*\WIL,radius=\KLRL]
arc[start angle=-\WIR,delta angle=2*\WIR,radius=\KLRR] ;
\end{tikzpicture}
\end{document}
```

Das LaTeX-Paket tikz-kalender, das auf TikZ aufbaut, ermöglicht den Ausdruck

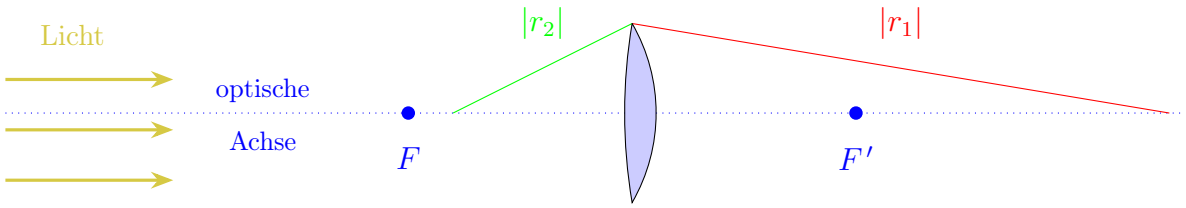


Abbildung 7.5: Krümmungsradien einer Sammellinse mit LaTeX (TikZ)

von Jahreskalendern. Man kann in einer besonderten Datei Einträge vorbereiten, die dann in den Kalender übernommen werden. Der folgende LaTeX-Code erstellt einen Jahreskalender für 2024 in schwarz-weiß, aber farbige Felder wären auch möglich.

```
\documentclass{tikz-kalender}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\setup{ year=2024, events={kalender},
  title={vivere militare est},lang=german,
  paper=a4, showweeknumbers, eventColor=white,
  titleColor=black, monthBGcolor=black,
  saturdayColor=gray!25, sundayColor=gray!35, }
\begin{document} \makeKalender \end{document}
```

Die Jahreszahl und den Titel, der als Überschrift dient, kann man natürlich ändern. Die Textdatei `kalender.events` kann mit einem einfachen Editor geschriebn werden. Um ihre Einträge in den Kalender zu übernehmen, sollte diese Datei im gleichen Verzeichnis wie das LaTeX-Dokument stehen. Ein Beispiel zeigt das Format der Einträge:

```
\event{\year-10-25}{Geb. Rotraud}
\event{\year-11-09}{Geb. Oma}
```

7.2.5 Sehr gutes Plotprogramm: Gnuplot

Ein Plotprogramm dient der graphischen Darstellung von mathematischen Funktionen und Messdaten auf einem ebenen Medium (Bildschirm oder Papier). Neben zweidimensionalen Objekten können auch dreidimensionale durch Projektion dargestellt werden.

Gnuplot ist ein leistungsfähiges Plotprogramm, das von der Kommandozeile, unter anderem in einer Linux-Konsole, gesteuert werden kann. Gnuplot-Befehle können leicht in einer Textdatei gespeichert werden. Damit ist es möglich, Gnuplot zusammen mit einer beliebigen Programmiersprache zu verwenden, auch mit der Shell.

Installation von Gnuplot

Wir installieren Gnuplot für die Konsole mit dem Befehl

```
sudo apt install gnuplot-nox
```

wobei nox (nicht für lateinisch Nacht, sondern) für *no X* steht. Für die Beschriftung der Graphiken sollten geeignete Fonts (Schriften) zur Verfügung stehen. Die Installation der entsprechenden Pakete wurde in Abschnitt 3.1.1 auf Seite 78 beschrieben.

Interaktiver Modus von Gnuplot

Der Befehl `gnuplot` startet den interaktiven Modus, der zum Üben geeignet ist. Er kann mit `q` (kurz für `quit`) wieder verlassen werden. Im interaktiven Modus gibt man Gnuplot-Befehle auf einer Kommandozeile ein, die mit dem Prompt `gnuplot>` beginnt. Eine direkte Ausgabe der Graphik in der Linux-Konsole ist behelfsmäßig möglich, indem mit Schriftzeichen eine Kurve geformt wird. Werden nacheinander die beiden Befehle

```
set terminal dumb ansi
plot sin(x)/x
```

eingegeben, erhält man einen Plot der Funktion $f(x) = sin(x)/x$ in der Konsole.

Gnuplot-Befehle in einer Textdatei, Ausgabe einer Bilddatei

Die Erzeugung einer graphischen Darstellung erfordert mehrere Gnuplot-Befehle, welche (wie der Quellcode einer Programmiersprache) mithilfe eines Editors in eine Textdatei (in unseren Beispielen: `gp.txt`) geschrieben werden. Die erzeugte Graphik wird in einer Bilddatei (in unseren Beispielen: `gp.png`) gespeichert. Diese kann in der Linux-Konsole mittels `fbi` oder `fim` im Framebuffer gezeigt werden (siehe Abschnitt 6.1.1 auf Seite 173). Das Format der produzierten Graphik wird durch Angabe eines „terminals“ bestimmt. In unseren Beispielen erzeugen wir mit `set terminal png` eine PNG-Graphik. Die Datei, in der die Graphik gespeichert wird wird als „output“ festgelegt, in unseren Beispielen mit `set output "gp.png"`. Zur Anzeige der Graphik dient dann der Befehl `fbi gp.png`.

Einfaches Beispiel: Sinusfunktion mit wenigen Befehlen

Folgender Code wird mit einem Editor in die Textdatei `gp.txt` geschrieben:

```
set terminal png
set output "gp.png"
a=2.5
f(x) = a*sin(x)
plot f(x)
```

Der Befehl `guplot gp.txt` erzeugt dann eine Datei `gp.png` mit der Darstellung der Funktion $f(x) = 2{,}5 \cdot \sin(x)$ als PNG-Bild der Größe 640×480.

Die ersten beiden Befehle des Codes wurden oben erläutert. Danach folgen die Wertzuweisung für eine Variable `a`, die Definition einer Funktion `f(x)` und den Befehl zur Zeichnung der Funktion. Die Parameter des Plots, zum Beispiel der Wertebereich der Funktion, werden hier von Gnuplot automatisch gesetzt. In den meisten Anwendungen wird man Befehle einsetzen, mit denen man die Plotparameter selbst bestimmt.

Einstellungen für „terminal“

Der Befehl `set terminal ...` bestimmt das Format der Graphik als „terminal“. Neben `png` gibt es weitere „terminals“. Der „terminal“ `png` erzeugt mithilfe der Programmbibliothek `libgd` eine Bilddatei vom Typ PNG. Standardmäßig gibt es dafür 256 Farben und die Größe (size) in Pixeln ist 640×480. Mit

```
set terminal png size 200,100
```

wird die Bildgröße auf 200×100 gesetzt. Die standardmäßige Option `nocrop` bestimmt, dass die Bildgröße fest ist, während `crop` Leerraum am Rand abschneidet und das Bild dadurch kleiner werden kann.

Nichtlineare Ausgleichsrechnung (Fit)

Für das folgende Beispiel wurden die Messpunkte

```
0.5 0.5
3 5
5.5 10
7 20
9.5 50
```

in einer Textdatei namens `m.txt` gespeichert. Die x-Werte stehen in der ersten, und die y-Werte in der zweiten Spalte. In jeder Zeile sind x- und y-Wert durch ein Leerzeichen getrennt. In eine Datei `gpfit` wurden folgende Gnuplot-Befehle geschrieben:

```
set terminal png
set output "gpfit.png"
set pointsize 3
f(x) = a*exp(b*x)+c
fit f(x) "m.txt" via a,b,c
set title sprintf("Fit mit f(x) = %.3f exp(%.3f x) + %.3f",a,b,c)
plot "m.txt" with points title"", f(x) with lines title "f(x)"
```

Der Befehl

```
gnuplot gpfit
```

führt die Gnuplot-Befehle der Datei `gpfit` aus. Sie passen eine Funktion $f(x) = a \cdot e\ b \cdot x + c$ an die Daten an (Fit), wobei die Parameter a, b und c von Gnuplot so gewählt werden, dass die Abweichung möglichst gering wird. Messwerte und angepasste Funktion werden dann in einer Graphik gezeichnet, die als Bilddatei `gpfit.png` gepeichert wird.

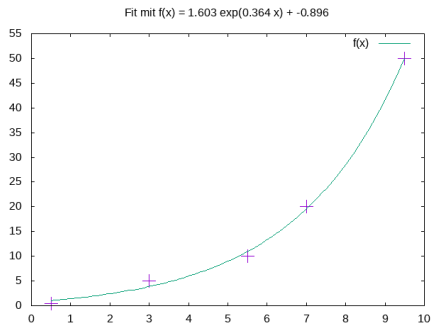


Abbildung 7.6: Fit mit `gnuplot`

7.2.6 Mächtige Mathematik: Mathematica

Das Programmpaket Mathematica der Firma Wolfram Research ist ein sehr mächtiges Werkzeug für *scientific computing*, einschließlich mathematischer Umformungen (Computeralgebra), wissenschaftlichen und technischen Berechnungen (Numerik) und deren graphische Darstellung (Visualisierung).

Nachteilig sind die Kosten für eine Lizenz und die gewöhnungsbedürftige Art des Programmierens. Andererseits verfügt Mathematica über sehr starke Befehle, mit denen manches schwierige Problem schnell gelöst und Bilder erzeugt werden können.

Mit dem Raspberry Pi darf man Mathematica zumindest teilweise kostenlos benutzen. Wenn man genug Speicherplatz (2,4 GB) hat, kann man es mit

```
sudo apt install wolfram-engine wolframscript
```

installieren. Auch ohne graphische Oberfläche können wir mit Mathematica arbeiten, aber der Raspberry Pi arbeitet langsamer als ein Supercomputer. Allerdings kann man, wenn man dem Raspberry Pi Zeit lässt, schwierige mathematische Aufgaben lösen.

Mathematica besteht aus einem Kernprogramm (kernel), das Berechnungen durchführt, und einer Benutzeroberfläche (front end). Die Benutzeroberfläche sieht auf verschiedenen Betriebssystemen unterschiedlich aus, aber das Kernprogramm hat immer die gleiche Funktionalität. Eingaben erfolgen in der Programmiersprache *Wolfram language*.

Die einfachste Möglichkeit, Mathematica anzusprechen, ist `wolframscript -c`, gefolgt von einer Fragestellung in Wolfram language. Zum Beispiel ergibt

```
wolframscript -c 2+3
```

nach einiger Zeit die richtige Antwort $\boxed{5}$. Statt `-c` kann man auch `-code` schreiben.

Um mit Mathematica interaktiv zu arbeiten, gibt man den Befehl

```
wolfram
```

ein und wartet bis der command prompt `In[1]:=` erscheint und Mathematica auf unsere Eingabe wartet. Die Zahl in der eckigen Klammer wird im Verlauf der Sitzung automatisch inkrementiert (hochgezählt) und ist meistens unwichtig. Nach unserer Eingabe

```
In[1]:= 2+3
```

und Drücken der ENTER-Taste erscheint die Antwort und ein neuer prompt.

```
Out[1]= 5
In[2]:=
```

Die mit `In[...]:=` oder `Out[...]=` eingeleiteten Bereiche heißen *input cell* beziehungsweise *output cell*. Die interaktive Schnittstelle arbeitet übrigens im Terminal schneller als in Betriebssystemen mit graphischer Oberfläche.

Zum Verlassen des Programms dient die Tastenkombination `Ctrl-D`.

Wir stellen nun die Syntax der Wolfram language in einfachen Beispielen vor:

Rechnen

In[1]:= 4.5 ∗ (2^2 - 2) / (1+1+2-1)
Out[1]:= 3.

Übliche arithmetische Operatoren ^ ∗ / + - ( ) werden verwendet und Ganzzahlen (`3`) von Fließkommazahlen unterschieden (`3.`). ^ steht für „hoch“ (Exponent).

In[2]:= Sqrt[2]
Out[2]:= Sqrt[2]

Funktionen erhalten ihre Argumente in eckigen Klammern. Da keine numerische Näherung verlangt wurde, wird das Ergebnis symbolisch ausgegeben.

In[3]:= Sqrt[2] // N
Out[3]:= 1.41421

Eine Näherung wird ausgegeben, wenn der Ausdruck in der input cell mit `//N` abgeschlossen wird und damit Argument der Funktion `N` ist (`N[Sqrt[2]]`).

In[4]:= N[Pi,20]
Out[4]:= 3.1415926535897932385

Das optionale zweite Argument der Funktion `N` gibt die Genauigkeit an, mit der das Ergebnis berechnet wird. Es gibt benannte mathematische Konstanten, z.B. `Pi`.

In[5]:= Im[1+2I] + I ∗ Re[1+2I]
Out[5]:= 2 + I

Komplexe Zahlen werden mit der mathematischen Konstanten `I` (Großschreibung!) gebildet. Die Funktionen `Re[ ]` und `Im[ ]` bilden den Real- und Imaginärteil.

In[6]:= 3 Abs[%]
Out[6]:= 3 Sqrt[5]

Das Zeichen % steht für das Ergebnis der letzten Berechnung. `Abs[ ]` bildet den Absolutwert einer komplexen Zahl. Eine Lücke ersetzt das Multiplikationszeichen ∗.

In[7]:= x = E Log[10,1000]
Out[7]:= 3 E

Das Zeichen `E` steht für die Eulersche Zahl e. `Log[a,b]` bildet den Logarithmus von `b` zur Basis `a`. Das Gleichheitszeichen weist der Variablen `x` einen Wert zu.

In[8]:= x = {1, 2, 3};

Geschweifte Klammern schließen eine geordnete Menge (*Liste*) von Elementen ein. Mit einem Semikolon abgeschlossene Ausdrücke erzeugen keinen Output.

In[9]:= Exp [x]
Out[9]:= {E, E^2, E^3}

`Exp [ ]` ist die Exponentialfunktion. Wenn man sie auf eine Liste anwendet, wird der Funktionswert von jedem Element gebildet. Die Variable `x` wird nicht geändert.

In[10]:= q = 3y; z = 2q + 5 q y
Out[10]:= 6 y + 15 y^2

Nicht nur numerische, auch symbolische Berechnungen sind möglich. Hier wird `z` berechnet. Durch ; getrennte Befehle oder Ausdrücke können in einer Zeile stehen.

Listen

```
In[11]:= y = {-1, 1 ,0 } ; xy = x . y
Out[11]:= 1
```

Der Variablen `x` wurde in input cell 8 eine Liste zugewiesen. Der Operator . bildet das Skalarprodukt (dot product) der Vektoren `x` und `y`. `xy` ist eine neue Variable.

```
In[12]:= x[[3]] + Part[y, 2]
Out[12]:= 4
```

Es gibt verschiedene Möglichkeiten, auf die Elemente einer Liste zuzugreifen. `x[[3]]` ergibt das dritte Element der Liste `x` und `Part[y, 2]` das zweite Element der Liste `y`.

```
In[13]:= y = Table[n!, {n,1,7,2}]
Out[13]:= {1, 6, 120, 5040}
```

`Table` erzeugt eine Liste von Funktionswerten der Variablen `n`, die Werte von 1 bis 7 in Schritten von 2 annimmt. `n!` oder `Factorial[n]` bildet die Fakultät von `n`.

```
In[14]:= Export[ "wurzeln.jpg", ListPlot[ Sqrt[y], PlotJoined->True ] ]
Out[14]:= wurzeln.jpg
```

`Sqrt[y]` erzeugt eine Liste, deren Elemente gleich der Quadratwurzel des entsprechenden Elements der Liste `y` sind. `ListPlot` erzeugt eine Graphik, in der die Elemente der eben erzeugten Liste gezeigt werden. Das erste Listenelement wird als Funktion von 1 dargestellt, das zweite als Funktion von 2, und so weiter. Die Option `Plotjoined->True` verbindet die Punkte der Grafik (Gegenteil: `Plotjoined -> False`).

Die Funktion `Export[ ]` speichert die Graphik in der Datei `wurzeln.jpg` im Bildformat JPEG* . (Aus unbekannten Gründen wird die Speicherung im Format PNG nicht unterstützt.)

Die Bilddatei kann mit einem Bildanzeigeprogramm, zum Beispiel `fbi` oder `fim`, auf dem Bildschirm angezeigt werden (siehe Abschnitt 6.1.1 auf Seite 173). Abbildung 7.7 zeigt die gespeicherte Graphik.

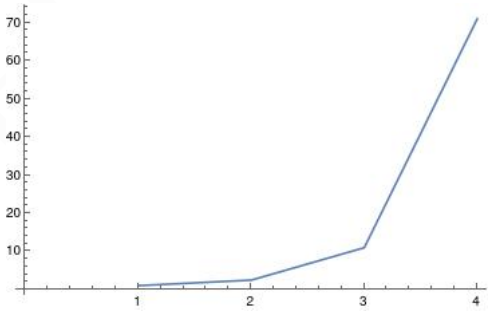


Abbildung 7.7: Plot der Funktion $f(x) = \sqrt{x}$ für $x \in \{1, 6, 120, 5040\}$

```
In[15]:= q = Array[a, 5]
Out[15]:= {a[1], a[2], a[3], a[4], a[5]}
```

`Array[ ]` erzeugt eine Liste symbolischer Elemente, die mit einem Index bis zur angegebenen Zahl (hier: 5) durchnummeriert sind. Die Indexwerte beginnen mit 1!

In[16]:= q=.; q=Array[b,{5,6},{0,1}]

Out[16]:= {{b[0, 1], b[0, 2], ...}}

`q=.` löscht die Variable `q` (eigentlich hier unnötig). Dann wird ein zweidimensionales Array `q` mit 5 mal 6 Elementen erzeugt, deren Indices von 0 bis 4 und von 1 bis 6 laufen.

Für aufwändigere Fragestellungen schreibt man in einem Texteditor wie `nano` ein Programm in der Wolfram language und speichert die Datei mit der Endung `.m` oder `.wl`, zum Beispiel als `mathe1.m`. In dem kurzen Beispielprogramm

```
(* Einfaches Programm mit Kommentar
und Textausgabe auf dem Bildschirm *)
Print["Hello world"]
```

steht ein zweizeiliger Kommentar und ein Befehl zur Ausgabe einer Zeichenkette. Ein- oder mehrzeilige Kommentare beginnen mir `(*` und enden mit `*)`. Man sieht, dass Zeichenketten (strings) in doppelten Anführungszeichen stehen. Funktionen, die in der Wolfram language bereits definiert sind (built-in functions), wie `Print`, beginnen mit einem Großbuchstaben. Funktionen, welche vom Nutzer definiert werden, sollten mit einem Kleinbuchstaben beginnen, um eine Verwechslung zu vermeiden. Man beachte, dass Argumente von Funktionen in eckigen Klammern stehen.

Zur Ausführung des Programms dienen der Befehl

```
wolframscript -f mathe1.m
```

und etwas Geduld. Statt `-f` kann man auch `-file` oder `-script` schreiben.

Will man das Ergebnis in eine Textdatei `erg.txt` schreiben, gibt man den Befehl

```
wolframscript -f mathe1.m > erg.txt
```

Wir schreiben nun eine Datei `mathe2.m` mit folgendem Quellcode:

```
t1 = {{-2, 10}, {0, 3}, {2, 2}}
g1 = ListPlot[t1, Prolog -> AbsolutePointSize[3]]
t2 = Table[{n, n*n - 3 n + 3}, {n, -3, 3, 0.1}]
g2 = ListPlot[t2, Joined -> True]
g3 = Show[g2, g1]
Export["mathe2.jpg", g3]
```

Die erste Befehlszeile erzeugt eine Tabelle mit Messwertpaaren. Die zweite Zeile zeichnet diese Tabelle (Liste) als Punkte im Koordinatensystem. Die dritte Zeile tabelliert die Funktion $f(n) = n^2 - 3 \cdot n + 3$, wobei der Definitionsbereich das Intervall [-3, 3] ist. Die vierte Zeile zeichnet die Tabelle (Liste) der Funktion und verbindet die Punkte. Die fünfte Zeile zeichnet beide Einzelgraphiken in einer neuen Graphik (`g3`), wobei der Bereich der x-Achse von der erstgenannten Einzelgraphik (`g2`) stammt. In der sechsten Zeile wird die Graphik `g3` im Bildformat JPEG gespeichert.

Die Ausführung der Datei `mathe2.m` mit dem Befehl

```
wolframscript -f mathe2.m
```

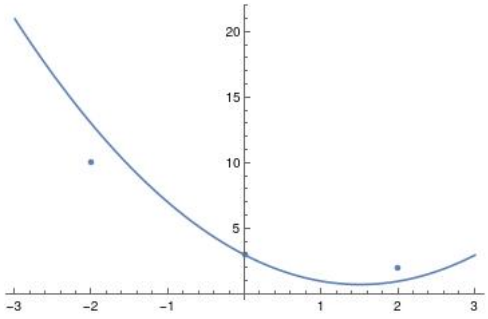


Abbildung 7.8: Überlagerung von zwei Plots `g2` und `g1` im Programm `mathe2.m`

erzeugt das in Abbildung 7.8 gezeigte Bild, gespeichert in der Datei `mathe2.jpg`.

Die mathematischen Visualisierungsmöglichkeiten zeigt ein Beispiel aus der Dokumentation von Mathematica (Wolfram Research, 1988, Plot3D, Wolfram Language function).

```
wolframscript -c Export["sin.jpg", Plot3D[Sin[x+y^2],{x,-3,3},{y,-2,2}]]
```

Die in der Bilddatei `sin.jpg` gespeicherte Graphik wird in Abbildung 7.9 gezeigt.

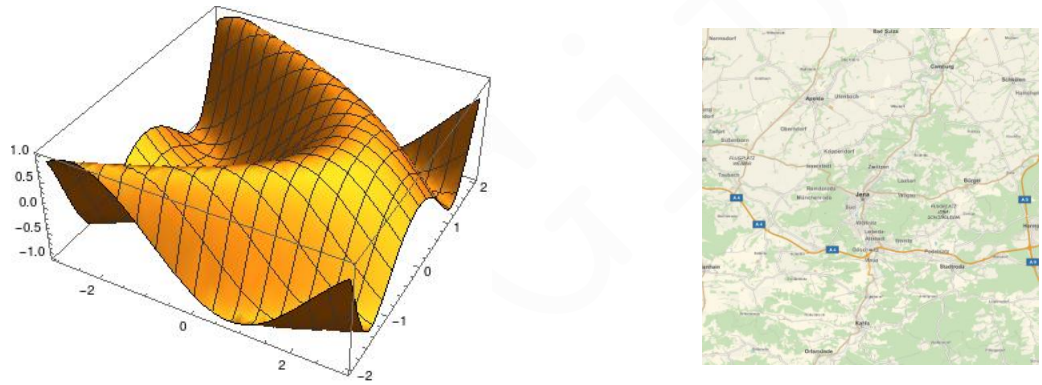


Abbildung 7.9: Graphiken mit Mathematica. Links: dreidimensionale Darstellung einer Funktion, rechts: Landkarte aus der Datensammlung von Mathematica.

Zur Demonstration des Umfangs von Mathematica führen wir folgenden Code aus.

```
wolframscript -c Export["jena.jpg", GeoGraphics[Jena]]
```

Wir erhalten eine Karte von Jena und Umgebung, siehe Abbildung 7.9.

Das folgende Programm speichert einen Kartenausschnitt der Größe 400 m × 400 m, dessen Zentrum bei den Koordinaten `{50.9285,11.5881}` liegt, in der Datei `jena.jpg`.

```
map = GeoGraphics[
{Red,PointSize[Large],
Point[GeoPosition[{50.92821,11.58884}]]},
```

```
GeoCenter->GeoPosition[{50.9285,11.5881}],
GeoRange->Quantity[200,"Meters"]
]
Export["jena.jpg", map]
```

Ein roter Punkt markiert den Ort mit den Koordinaten `{50.92821,11.58884}`.

Man kann auch mit physikalischen Größen und Konstanten rechnen. Das Programm

```
d = PlanetData["Earth","DistanceFromSun"]
v = Quantity["SpeedOFLight"]
t = d / v
Print[ UnitConvert[t,"Minutes"] ]
```

berechnet die Zeit, welche das Licht von der Sonne zur Erde benötigt, und gibt das Ergebnis in Minuten aus. Die Syntax der Größen und Einheiten ist leider schlecht.

Mehr über die Wolfram language erfährt man in der online Dokumentation:

https://www.wolfram.com/language/elementary-introduction/2nd-ed/

https://reference.wolfram.com/language/

8 Python

Mit Python als mächtiger, gut unterstützter Programmiersprache kann man viele besondere Aufgaben lösen und in Linux-Betriebssystemen ohne graphische Oberfläche Defizite ausgleichen, die aus der fehlenden Weiterentwicklung von Dienstprogrammen für Betriebssysteme ohne graphische Oberfläche entstanden. In den meisten Linux-Distributionen, auch im Raspberry Pi OS, ist Python bereits installiert. Aufgrund der Weiterentwicklung kann es wichtig sein, die aktuelle Version zu kennen, die man mit dem Befehl `python --version` erfährt.

8.1 Kurze Einführung

8.1.1 Einleitung

Warum Python?

Python ist eine universelle Programmiersprache. Das heißt, es läuft auf allen wichtigen Betriebssystemen, ist für verschiedene Anwendungen geeignet und auf vielen Hardwareplattformen verfügbar. Python ist eine höhere Programmiersprache, da es leicht zu lesen ist und besondere, komplexe Strukturen enthält, welche einen kurzen Quellcode ermöglichen. Python ist effizient und leistungsfähig, da es Teile (Module) bereitstellt, die (in C oder C++ programmiert) besonders schnell arbeiten und ständig weiterentwickelt werden. Python ist vergleichsweise leicht zu erlernen, gut dokumentiert und, last but not least, für den normalen Nutzer kostenlos.

Wie schreibt man Python-Code?

Python ist eine Programmiersprache, deren Quellcode übersichtlich strukturiert ist und über ein Interpreterprogramm sofort ausgeführt werden kann (Skriptsprache). Es gibt eine umfangreiche Standardbibliothek für Unterprogramme und viele Erweiterungen (Bibliotheken und Module) für besondere Aufgaben.

Es wird empfohlen, die neueste stabile Version der 3er-Serie (Python 3), kurz Python, zu verwenden. Das ältere Python 2 wird hier nicht berücksichtigt.

Python-Code kann in einer integrierten Entwicklungsumgebung (IDE) oder in einem einfachen Texteditor geschrieben werden. Für Betriebssysteme mit graphischer Oberfläche gibt es integrierte Entwicklungsumgebungen, zum Beispiel IDLE (einfach) oder Spyder. In der Linux-Konsole schreibt man Programme mit einem Texteditor, zum Beispiel `nano` (siehe Seite 131). Einige Texteditoren, wie `vim` (siehe Seite 133) und `emacs`, können die Funktion von Entwicklungsumgebungen weitgehend übernehmen.

Ein Programmierparadigma beschreibt schlagwortartig den Programmierstil. Es gibt verschiedene Definitionen (und darauf aufbauende Klassifikationen einzelner Programmiersprachen) und fließende Übergänge. Eine einfache Einteilung ist

Imperative Prorgammierung (im engeren Sinne)
(maschinennahe Urform, Kette direkter Befehle mit Sprunganweisungen)

Imperative Prorgammierung (im weiteren Sinne)
(Befehlskette mit einfachen Blockstrukturen, wie Schleifen und Unterprogrammen

Strukturierte Prorgammierung
(stark geordnete Zusammenfassung von Befehlen und wiederverwendbare Blöcke)

Zur strukturierten Programmierung gehören insbesondere

Prozedurale Programmierung
(Kapselung von Code in eigenständige Unterprogramme oder Funktionen)

Objektorientierte Programmierung
(Objekte als formalisierte logische Teile, deren Muster Klassen sind)

Deklarative Programmierung
(Beschreibung des Ziels einer Berechnung unabhängig vom Rechenweg)

Obwohl die Elemente von Python durch Objekte, auch Instanzen genannt, gebildet werden, sind neben objektorientierten auch prozedurale Programmierparadigmen möglich. Die vorliegende Skripte meidet objektorientierte Programmierung und bevorzugt kurze Beispiele mit imperativer Programmierung im weiteren Sinne und mit funktionaler prozeduraler Programmierung.

Python kann in einem interaktiven Modus in der Linux-Konsole benutzt werden (oder in einem anderen Terminal von Betriebssystemen mit graphischer Oberfläche). Man startet ihn mit dem Befehl

```
python -i
```

Zum Verlassen dient die Tastenkombination `Ctrl-D` (siehe Seite 13). Der interaktive Modus ist nützlich, wenn man Python lernt. Mehr dazu findet man zum Beispiel auf der Webseite https://openbook.rheinwerk-verlag.de/python/03_001.html

Erstes Beispiel: Hallo Welt

Bei der Vorstellung einer Programmiersprache ist es üblich, mit einem Programm zu beginnen, das den einfachen Text „Hallo Welt!“ ausgibt. Folgendes Programm ist eine Erweiterung, die noch mehr zeigt. Es wird mit einem Texteditor geschrieben und in der Datei `program.py` gespeichert.

```
1 #!/usr/bin/env python3
2 # erweitertes Hallo-Welt-Programm zur Begrüßung
3 print("Hallo Welt!") # das war einfach
4 text = input("Ihr Name: ")
5 if text:
6     print("Hallo "+text)
```

Die erste Zeile ist nur für wenige Betriebssysteme sinnvoll. Andere Betriebssysteme beachten sie nicht, da sie mit einer Raute # am Anfang als Kommentar gekennzeichnet ist. In der Regel kann man diese Zeile weglassen. Wenn man jedoch ein Linux-Betriebssystem hat, welches nicht weiß, wie es Python-Programme ausführen soll, ist sie hilfreich. Diese merkwürdige erste Zeile hat sogar einen eigenen Namen: Shebang-Zeile. Da wir die Shebang-Zeile vermutlich nicht brauchen, lassen wir sie ab jetzt meistens weg.

Mit oder ohne Shebang-Zeile wird das Programm durch

```
python3 program.py
```

ausgeführt.

Mit Shebang-Zeile kann man das Programm auch mit einem kürzeren Befehl ausführen lassen: Nehmen wir an, wir befinden uns im Verzeichnis mit dem Pythonprogramm `program.py`. Das Programm wird mit

```
chmod a+x program.py
```

als ausführbar gekennzeichnet und kann nun mit

```
./program.py
```

gestartet werden. Aufgrund der Shebang-Zeile müssen wir im Befehl nicht `python3` schreiben. Andererseits müssen wir bei der Benutzung des Befehls `python3 program.py` die Datei `program.py` nicht als ausführbar gekennzeichnet haben.

Wenn `python3` im Verzeichnis `/usr/bin/` liegt, wie bei unserem Betriebssystem, dann ist auch eine Shebang-Zeile `#!/usr/bin/python3` richtig. Der Vorteil der oben benutzten Shebang-Zeile `#!/usr/bin/env python3` ist, dass das Hilfsprogramm `env`, welche das Programm `python3` findet, *immer* im Verzeichnis `/usr/bin/` liegt.

Die zweite Zeile des obigen Programms beginnt mit einer Raute # und ist daher ein Kommentar. Wer den Programm-Code liest, erfährt in Kürze, was das Programm tut.

In der dritten Zeile wird mit der Funktion print() ein Text, genauer: eine Zeichenkette (String) ausgegeben. Funktionen können Werte entgegennehmen, die in runden Klammern eingefasst sind. Ein an die Funktion übergebener Wert heißt Argument. Zeichenketten stehen entweder in doppelten oder in einfachen geraden Anführungszeichen. Nach der Raute # folgt ein Kommentar, der bis zum Zeilenende reicht.

In der dritten bis sechsten Zeile stehen Anweisungen (englisch: statements), mit denen wir dem Python-System etwas zu tun geben. Wir sagen statt Anweisung auch Befehl (command), weil das im Deutschen ein kürzeres Wort ist. Anweisungen werden meistens von links nach rechts abgearbeitet. Man beachte aber, dass eine Zuweisungsanweisung mit mehr als einem Zuweisungssymbol = von rechts nach links ausgewertet wird. $x = y = z$ entspricht $x = (y = z)$.

Am Anfang der vierten Zeile steht der Name einer Variablen, nämlich `text`. Damit wird ein Objekt mit diesem Namen geschaffen. Ein Objekt, das Speicherplatz belegt, heißt in Python Instanz. Der Name heißt Referenz. Der Referenz `text` folgt das Symbol der Zuweisung =, mit dem unserem Objekt ein Wert zugewiesen wird. Rechts vom Zuweisungssymbol steht ein Ausdruck, der eine Zeichenkette erzeugt. Ein Ausdruck (englisch

expression) ist etwas, das einen Wert ergibt. Streng genommen gibt es in Python keinen Zuweisungsoperator, wie etwa in C. Das Symbol = ist nur ein Teil der Zuweisungsbefehls (assignment statement) gemäß dessen Syntax (Schreibweise).

Die Funktion input() erhält als Argument eine Zeichenkette und zeigt sie im Terminalfenster. Dann wartet sie, bis der Nutzer eine neue Zeichenkette über die Tastatur eingegeben und mit der Return-Taste abgeschlossen hat. Diese Zeichenkette ist der Wert, der `text` zugewiesen wird.

In der fünften und sechsten Zeile steht eine bedingte Anweisung. Der Befehl in der fünften Zeile prüft, ob der dem `if` folgende Ausdruck, hier `text`, einen Wert ergibt, der wahr oder unwahr ist. Was Wahrheit genau bedeutet, besprechen wir später. Hier wird eine leere Zeichenkette (wenn der Nutzer nur die Return-Taste gedrückt hat) als unwahr gewertet und alles andere (wenn der Nutzer eine Zeichenkette eingegeben hat) als wahr. Am Ende der Prüfung muss ein Doppelpunkt `:` stehen!

In der sechsten Zeile steht ein Befehlsblock, der nur ausgeführt wird, wenn die vorige Prüfung wahr ergab und sonst eben nicht. Gegebenenfalls wird mit der Print-Funktion eine Zeichenkette ausgegeben. Ein Befehlsblock ist eine zusammenhängende Abfolge von Befehlen. In diesem Beispiel besteht der Block allerdings aus nur einem Befehl. Einen Befehlsblock erkennt man daran, dass alle enthaltenen Befehlszeilen mit der gleichen Anzahl an Leerzeichen eingerückt sind; hier mit vier Leerzeichen. Die Zeichenkette besteht aus zwei Teilen, erstens `"Hallo "` und zweitens der Wert von `text`. Verbunden werden die beiden Teile durch den sogenannten Konkatenationsoperator `+`. Das Symbol `+` steht in diesem Zusammenhang also nicht für den Additionsoperator, sondern den Konkatenationsoperator. Letzterer fügt zwei Zeichenketten zu einer neuen zusammen.

Wenn man als Name *Susanne* eingibt, erscheint die Ausgabe `Ihr Name: Susanne` und, in der nächsten Zeile, `Hallo Susanne`.

Virtual Environments

Einige neue Linux Betriebssysteme, einschließlich das hier verwendete (auf Debian Bookworm beruhende), haben Python als „externally managed“ markiert, sodass nur die Paketverwaltung des Betriebssytems Änderungen vornehmen darf.

Damit funktioniert das Paketverwaltungsprogramm `pip3` nicht, außer wenn man die Option `-break-system-packages` am Ende des Paketinstallationsbefehls von `pip3` anfügt. Letzteres bedeutet: Der Nutzer übernimmt das Risiko , dass ein so installliertes neues Paket das Pythonsystem (zer)stört. Dieses Risiko ist vielleicht nicht groß, aber falls irgendetwas schief geht, muss man untersuchen, ob nicht doch ein neues Paket schuld ist. Wer vorsichtig oder feige ist, sollte für Pythonprojekte, in denen Pakete über `pip3` installiert werden, ein *virtual environment* verwenden.

Ein *virtual environment* ist ein besonderes Verzeichnis, in dem die eigenen Module eines Pythonprojekts gespeichert sind und wo es erlaubt ist, mit `pip3` Pakete zu installieren. Geht etwas schief, ist nur das Projektverzeichnis betroffen. Für virtual environments braucht man das Modul `venv`. Es wird durch

```
sudo apt install python3-venv
```

installiert (siehe Seite 78). In der Regel wird für jedes Pythonprojekt ein eigenes virtual environment angelegt. Vereinfachend wollen wir hier nur eines namens `mypy` schaffen.

```
cd ~/progs
python3 -m venv mypy
```

Ein Beispiel für den Gebrauch eines virtual environments findet man auf Seite 391.

8.1.2 Schreibweisen, Schlüsselwörter, Blöcke, Instanz, Referenz

Normalerweise steht auf einer Zeile nur ein Befehl. Zwei oder mehr Befehle, die normalerweise ohne Einrückung oder mit gleicher Einrückung (Blockstufe, siehe unten) in aufeinander folgenden Zeilen geschrieben werden, können auch in einer Zeile stehen, wenn zwischen ihnen ein Semikolon gesetzt wird.

Das Dezimaltrennzeichen für Gleitkommazahlen (rationale Zahlen, die mit endlich vielen Ziffern dargestellt werden) ist, wie in den USA üblich, ein Punkt und kein Komma.

```
x = 3.1415926
```

Eine langer Befehl kann auf zwei Zeilen verteilt werden. Am Ende der ersten Zeile wird ein Backslash \ gesetzt. Die zweite Zeile setzt den Befehl ohne Einrückung fort.

```
print(Erstens kommt es anders, zweitens\
als man denkt.)
```

Schlüsselwörter (key words, reserved words) sind Namen oder Bezeichner (siehe unten), die eine besondere Bedeutung in der Programmiersprache haben. Sie können daher nicht als Bezeichner für Variablen verwandt werden. Schlüsselwörter in Python sind

and, as, assert, async, await, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, while, with, yield

`False`, `None` und `True` werden groß geschrieben, die anderen klein. Ob ein Bezeichner ein Schlüsselwort ist, wird mit der Funktion iskeyword() des Moduls keyword geprüft.

```
1 # Beispiel zu Schlüsselwörtern
2 import keyword
3 w = "for"
4 if keyword.iskeyword(w): print(w, "ist Schlüsselwort.")
5 else: print(w, "ist kein Schlüsselwort.")
```

Die Liste `keyword.kwlist` enthält (als Zeichenketten) alle Schlüsselwörter.

In Python werden Namen für Objekte auch *Bezeichner* genannt. Die Zeichenkette eines Bezeichners muss mit einem Buchstaben des englischen Alphabets (ohne ß und ohne Umlaute wie Ä oder ü) oder einem Unterstrich _ beginnen. Danach sind auch Ziffern erlaubt. Groß- und Kleinbuchstaben werden unterschieden.

Bezeichner müssen eindeutig sein. Wenn das Hauptprogramm (welches ein Modul ist, das Hauptmodul) andere Module oder anderen Code lädt, muss darauf geachtet werden,

dass der gleiche Bezeichner nicht doppelt im Hauptprogramm und verschiedenen Modulen vergeben wird. Deshalb gibt es *Namensräume* (namespaces), die den Geltungsbereich eines Bezeichners einschränken. Der Namensraum des Hauptprogramm heißt *globaler Namensraum.*

Jedes Modul hat einen eigenen Namensraum. Zum Beispiel gibt es im Modul `math` den Bezeichner `sin` für die Sinusfunktion von Winkeln im Bogenmaß. Nach dem Laden des Moduls mit `import math` kann man mit `math.sin(math.pi)` den Wert von $\sin(\pi)$ berechnen. Das Hauptprogramm kann eine eigene Funktion `sin` enthalten, die mit `math.sin` nicht verwechselt wird. Man kann den Namensraum eines Moduls in den globalen Namensraum einbinden, zum Beispiel mit `from math import *`. Dann werden bereits vorhandene, gleichlautende Bezeichner überschrieben und man ruft zum Beispiel die Sinusfunktion mit `sin(pi)` ohne den Vorsatz `math.` auf.

Jede Funktion (siehe unten) hat einen eigenen Namensraum. Die Bezeichner einer Funktion (für lokale Referenzen) werden ohne Vorsatz des Funktionsnamens verwendet. Es kann gleichlautende Bezeichner im globalen Namensraum geben (für globale Referenzen). Python unterscheidet die Bezeichner danach, ob Sie innerhalb oder außerhalb des Funktionskörpers vorkommen.

Ein oder mehrere Befehle können zu einem Block zusammengefasst werden. Befehlsblöcke werden in Kontrollstrukturen und Funktionsdefinitionen gebraucht (siehe unten). Ein Block wird durch Einrückung (indentation) der Befehle in ihrer Zeile gebildet. Üblich ist die Einrückung mit 4 Leerzeichen, aber eine andere konstante Anzahl an Leerzeichen ist auch möglich. Tabulatoren (Tabs) sollten nicht dafür verwendet werden. Blöcke können verschachtelt werden. Folgendes Beispiel enthält zwei Blockstufen.

```
1 # Beispiel zu Blockstufen
2 if x != 0:
3     for i in [1,2,3]:    \# eine Blockstufe
4         print(i/x)       \# zwei Blockstufen
```

Eine *Instanz* ist, im Sinne der objektorientierten Programmierung, ein im Rechner gespeicherter Wert von bestimmtem Typ. Jede Instanz hat also einen Datentyp, einen Wert und einen bestimmten Speicherplatz. Letzterer wird durch Angabe einer Identität genannten Nummer beschrieben. Zwei Instanzen können den gleichen Datentyp und den gleichen Wert haben, aber sie unterscheiden sich immer in der Identität.

Eine *Referenz* ist der Name einer Instanz. In der Mathematik bildet eine Variable einen Namen für verschiedene mögliche Werte. Entsprechend bildet bei Python eine Referenz den Namen, welchem verschiedene Instanzen zugewiesen werden können. Eine Referenz ist also ein Verweis auf eine Instanz, der beliebig (auch mit unterschiedlichem Datentyp) geändert werden kann. In der Befehlsfolge

```
a = 12.3
b = a
a = 'qwertz'
```

wird zunächst der Referenz a eine Instanz mit dem Wert `12.3` und dem Datentyp float zugewiesen. Die Instanz hat auch eine eindeutige Identität, deren Zahlenwert hier allerdings unbekannt ist. In der zweiten Zeile wird der Referenz b dieselbe Instanz zugeordnet wie der Referenz a. Jetzt verweisen also zwei Referenzen auf dieselbe Instanz. In der dritten Zeile wird der Referenz a eine andere Instanz mit dem Wert `'qwertz'` und dem Datentyp str zugewiesen. Nun verweisen a und b auf verschiedene Instanzen.

Die Funktion id() gibt die Identität der im Argument genannten Instanz zurück.

`print( id(a) )` ergibt eine Ganzzahl (`int`) wie `140396484265496`

Einiges wird in vorliegendem Manuskript vereinfacht dargestellt. Viele Objekte, die sich wie Funktionen verhalten, sind eigentlich Klassen[1]. Wenn wir ein Objekt mit runden Klammern aufrufen, ist es ein `callable` und das kann eine Funktion sein (der Argumente für Funktionsparameter übergeben werden) oder eine Klasse (der Argumente zur Bildung einer Instanz übergeben werden). Beispielsweise sind, streng genommen, `int("42")` und `range(10)` keine Funktionsaufrufe. Aber, `int` und `range` sehen aus wie Funktionen und verhalten sich wie Funktionen, also nennen wir sie Funktionen.

8.1.3 Datentypen

Alle Daten haben einen Typ. *Numerische Datentypen* enthalten jeweils eine Instanz (eine Zahl). Es gibt auch einen Datentyp *NoneType*, der das Fehlen von Daten ausdrückt.

Datentypen, die mehrere Instanzen zusammenfassen, heißen containers oder collections. Zu ihnen gehören sets (Mengen), sequences (Folgen) und mappings (Zuordnungen). Die darin enthaltenen Instanzen nennt man items oder elements (Elemente). Jeder container, der bereits existierende Elemente enthält, hat eine Methode `__contains__`.

Weitere container-Datentypen, zum Beispiel `array` und `collections`, werden durch Module bereitgestellt. Streams (Datenströme) sind ein besonderer container-Datentyp, dessen Elemente sich nicht im Speicherraum verteilen, sondern in der Zeit. Sie entspringen einer source (Quelle) und enden in einem sink (Ziel). Zum Beispiel können Dateien oder Netzwerkverbindungen als source oder sink dienen.

Darüber hinaus kann man auch eigene Datentypen definieren.

Die Funktion type() gibt den Datentyp der Instanz zurück, die ihr im Argument als Literal oder Referenz übergeben wurde. Im folgenden Beispiel bezeichnet das Schlüsselwort `True` eine Instanz des Datentyps `bool` (siehe unten).

`type(True)\\[2mm]` ergibt `<class 'bool'>`

`type([1, 2, 3])` ergibt `<class 'list'>`

Die Datentypen bilden Klassen, im Sinne der objektorientierten Programmierung. Die Funktion type() gibt den Datentyp des Arguments daher als Angabe einer Klasse zurück.

type unterscheidet Datentypen einer Unterklasse (zum Beispiel *Counter*), die durch Vererbung aus einer Oberklasse (zum Beispiel *dict*) entstanden sind. Dagegen ist der

[1] für einen Rosinen ausscheidenden Pythontheoretiker

Rückgabewert des Funktionsaufrufs `isinstance(collections.Counter([1,2,3,2]), dict)` `True`, da *Counter* von *dict* erbt.

Manchmal muss eine Instanz durch eine andere Instanz mit ähnlichem Wert, aber anderem Datentyp ersetzt werden. Hierzu dienen Funktionen, deren Namen sich vom gewünschten Datentyp ableiten. Zum Beispiel wandelt `int(12.3)` die Gleitkommazahl 12.8 in die Ganzzahl 12 um. Dabei werden Nachkommastellen abgeschnitten (also nicht gerundet). `float(33)` wandelt die Ganzzahl 33 in eine Gleitkommazahl 33.0 und `str(b)` die von `b` referenzierte Instanz (Beispiel: `12.8`) in eine Zeichenkette (hier: `'12.8'`).

Numerische Datentypen

- int (Ganzzahlen), z.B. `11`
- float (Gleitkommazahlen), z.B. 3.1416
- bool (logische Werte = boolesche Werte) `True` und `False`
- complex (komplexe Zahlen), z.B. `3.5 + 4j`

Vom Datentyp bool (nach G. Boole) gibt es nur zwei Werte, von anderen numerischen Datentypen „unendlich“ viele, womit gemeint ist: so viele, wie der Computerspeicher zur Verfügung stellen kann. Instanzen der numerischen Datentypen sind immutabel, das heißt: nach einer Veränderung des Werts (Rechnung) wird eine neue Instanz erzeugt.

Sequenzielle Datentypen

- str (Zeichenketten, strings genannt), z.B. `'Hallo'`
- list (veränderliche Liste), z.B. `['eins', 2, 3.0]`
- tuple (unveränderliche Liste), z.B. ("xy", 2, 3.0)
- bytes (Binärdaten, immutable Byte-Sequenz)
- bytearray (Binärdaten, mutable Byte-Sequenz)

Sequenzielle Datentypen enthalten eine Folge von Elementen, die beginnend mit dem Index 0 abgezählt werden. Ist `s` die Instanz eines sequenziellen Datentyps, gibt `s[5]` das Element mit dem Index 5 zurück (das sechste Element). Mit `s[1:5]` erhält man die Teilsequenz, die mit dem Index 1 (das zweite Element) beginnt und mit dem Index 5 endet (das Element mit dem Index 6 wird nicht mehr genommen).

Ein `tuple` (deutsch: Tupel) ist eine Zusammenfassung beliebig vieler Objekte von beliebigem (auch unterschiedlichem) Typ in einer bestimmten Reihenfolge. Ein tuple ist immutable (unveränderlich), das heißt: bei Veränderungen wird ein neues Objekt vom Datentyp tuple erzeugt.

Durch Kommata getrennte Objekte erzeugen eine Instanz vom Datentyp tuple, wenn sie von runden Klammern eingefasst werden oder ohne Klammern dastehen. Im Befehl

```
x = "Sinn", 42
```

wird ein tuple aus zwei Elementen (Paar) gebildet. Das Komma wird, anders als in der Programmiersprache C, nicht als Aufzählungsoperator betrachtet, sondern erzeugt

in der Regel einen sequentiellen Datentyp (abhängig vom verwendeten Klammertyp).[2] Die Einfassung in runde Klammern ist zum Beispiel wichtig, wenn ein tuple in den runden Klammern einer Funktion vorkommt, damit man es von drei einzelnen Instanzen unterscheiden kann. Nach dem letzten Element eines Tupels darf man ein Komma setzen, muss es aber nicht, außer bei einem einelementigen Tupel. Ein Tupel mit nur einem Element braucht ein Komma, wie in `("A",)`, um es von einer einzelnen Instanz zu unterscheiden, die in runden Klammern stehen darf, zum Beispiel `("A")` (kein Tupel).

Auch eine `list` (deutsch: Liste) ist eine Zusammenfassung beliebig vieler Objekte von beliebigem (auch unterschiedlichem) Typ in einer bestimmten Reihenfolge. Eine list ist aber mutable (veränderlich), das heißt: bei Veränderungen wird kein neues Objekt erzeugt.

Durch Kommata getrennte Objekte erzeugen eine Instanz vom Datentyp list, wenn sie von eckigen Klammern eingefasst werden. Im Befehl

```
x = [ "Sinn", [2, 3, 7] ]
```

wird eine list aus zwei Elementen gebildet, deren zweites Element eine list ist.

Tabelle 8.1 führt Operatoren und Methoden für alle sequenziellen Datentypen auf und Tabelle 8.2 weitere Methoden für lists. Zeichenketten (strings) werden im Abschnitt 8.1.7 behandelt.

`A + B`	Konkatenation von `A` und `B`	`[1, 4,] + [9]` → `[1, 4, 9]`
`x in A`	`True` wenn `x` ∈ `A`, sonst False	`2 in (1, 4, 9)` → `False`
`x not in A`	`True` wenn `x` ∉ `A`, sonst False	`"u" not in "Jena"` → `True`
`A[i]`	das Element von `A` mit Index `i`	`"Jena"[1]` → `'e'`
`A[i:j]`	die Elemente mit Indices `i` bis `j-1`	`"Jena"[1:3]` → `'en'`
`len(A)`	Anzahl der Elemente von `A`	`len("Jena")` → `4`
`A.index(x)`	Index von erstem `x` in `A`	`"Pfeiffer".index("f")` → `1`
`A.count(x)`	Häufigkeit von `x` in `A`	`"Pfeiffer".count("f")` → `3`
`min(A)`	kleinstes Element (mit Ordnung)	`min( [1, 2, 3] )` → `1`
`max(A)`	größtes Element (mit Ordnung)	`max(["a","b","c"])` → `'c'`

Tabelle 8.1: Operatoren und Methoden für sequenzielle Datentypen. `min` und `max` setzen eine Ordnungsrelation voraus, die auf alle Elemente angewandt werden kann.

NoneType

- `None`

Dieser Datentyp hat nur eine Instanz, nämlich `None`. Diese ist ein Platzhalter, wenn andere Instanzen fehlen oder ausfallen. `None` kann nicht addiert oder konkateniert werden.

[2] „In der Regel“ heißt hier, dass Python intern, um den Code schneller auszuführen, manchmal auf die Erzeugung einer neuen Instanz verzichtet, aber das merkt der Programmierer nicht.

`L.append(x)`	Anhängen von `x` am Ende	`L=[6,7];L.append(8)` → `[6, 7, 8]`
`L.extend(M)`	Konkatenation von `L` und `M`	`L=[1]; L.extend( [2] )` → `[1, 2]`
`L.insert(i,x)`	Einfügung von `x` an `i`	`L=[9] ; L.insert(0,8)` → `[8, 9]`
`x = L.pop()`	entnimmt letztes Element	`L=[3, 4, 5] ; L.pop()` → `[3, 4]`
`L.remove(x)`	Entfernung von erstem `x`	`L=[6,7,6];L.remove(6)` → `[7, 6]`
`L.reverse()`	Reihenfolge umgekehrt	`L=[1,2] ; L.reverse()` → `[2, 1]`
`L.sort()`	Sortierung (mit Ordnung)	`L=[3,1,2];L.sort()` → `[1, 2, 3]`

Tabelle 8.2: Methoden für lists (zusätzlich zu den in Tabelle 8.1 genannten). `pop` gibt das entnommene Element zurück. Im Beispiel wird `x` der Wert 5 zugewiesen.

Zuordnungen (mappings)

- dict (dictionary), z.B. {" aga" : "R", "ata" : "I" }

Ein dictionary (deutsch: Wörterbuch) ist eine Menge, die aus Zuordnungen besteht, wobei jeweils einem Schlüssel (englisch: key) ein Wert (value) zugeordnet wird. In obigem Beispiel sind Schlüssel und Werte Zeichenketten, aber es können auch andere Datentypen sein. Für Schlüssel sind jedoch nur unveränderliche (immutable) Datentypen erlaubt.

dict ist ein mutabler Datentyp: Der Inhalt des dictionary kann verändert werden, ohne dass eine neue Instanz gebildet wird. Folgendes Beispiel gibt `Schlüssel: ata` und `Wert: I` aus.

```
1 # Beispiel zum Datentyp dictionary
2 d = { "aga" : "R", "ata" : "I" } # dictionary definieren
3 print("Schlüssel:", "ata"))       # key = Schlüssel
4 print("Wert:", d["ata"])          # value = Wert
```

Erfolgt der Zugriff auf den Wert des Schlüssels `k` eines dictionary d durch `d[k]`, tritt ein Fehler auf, wenn es den Schlüssel `k` nicht gibt. Es ist oft besser, wenn in diesem Fall ein Standardwert (englisch: default value) als Ersatzwert zurückgegeben wird. Das wird mit der Methode `get` erreicht. Das folgende Beispielprogramm

```
1 d = {1 : one, 2 : "two"}
2 k = d.get(3, "not found")
3 print(k)
```

gibt `not found` auf dem Bildschirm aus.

In einem dictionary `d` wird durch den Befehl

```
d[1] = "uno"
```

ein Schlüssel `1` mit dem Wert `"uno"` erzeugt. Falls es den Schlüssel `1` bereits gab, wurde das alte Schlüssel-Wert-Paar damit überschrieben. Mit dem Methodenaufruf

```
d.setdefault(1, "uno")
```

wird ein neues Schlüssel-Wert-Paar nur dann erzeugt, wenn es den Schlüssel `1` noch nicht gibt. Die Methode gibt jedenfalls den Wert des Schlüssels `1` zurück, aber der wird hier nicht verwendet. Das folgende Beispielprogramm

```
1 d = {1 : one, 2 : "two"}
2 d[1] = "uno"
3 d.setdefault(2, "dos")
4 d.setdefault(3, "three")
5 print(d)
```

gibt `{1: 'uno', 2: 'two', 3: 'three'}` aus.

Gegeben sei ein dictionary `d`, dessen Werte Listen sind. Nun soll die dem Schlüssel `k` zugeordnete Liste `d[k]` am Ende mit dem Wert `x` erweitert werden. Das ist aber nur möglich, wenn es den Schlüssel (und die zugehörige Liste) bereits gibt. Daher wollen wir, falls es den Schlüssel `k` noch nicht gibt, ihn erzeugen und ihm eine leere Liste zuordnen, bevor die (leere) Liste mit dem Wert `x` erweitert wird. Der einzeilige Befehl

```
d.setdefault(k, [ ]).append(x)
```

macht genau das. Das folgende Beispielprogramm

```
1 d = {1 : ["a", "b"], 2 : ["o", "p"]}
2 d.setdefault(1,[]).append("c")
3 # print(d)
4 d.setdefault(3,[]).append("x")
5 print(d)
```

gibt somit `{1: ['a', 'b', 'c'], 2: ['o', 'p'], 3: ['x']}` aus.

Die Schlüssel und Werte in einem dictionary können auf sogenannte *view objects* abgebildet werden. Anders als bei einer Liste, kann man auf die Elemente des dictionary, die im view object abgebildet werden, nicht mit einem Index zugreifen (not subscriptable). Das view object ist jedoch iterierbar und kann effizient in einer for-Schleife (siehe unten, Seite 277) durchlaufen werden, ohne dass eine Liste gebildet wird. Vorteilhaft ist, dass ein view object automatisch aktualisiert wird, wenn das dictionary verändert wird.

Die Methoden `keys` und `values` erzeugen ein view object der Schlüssel beziehungsweise der Werte. Das view object `items` bildet alle Schlüssel, zusammen mit dem jeweils zugehörigen Wert, als Paare ab. Folgendes Beispiel zeigt die automatische Aktualisierung des view objects von `items` und die Verwendung in einer for-Schleife.

```
1 d = {"one":"eins", "two":"zwo"}
2 v = d.items()      # view object for keys and values
3 d["two"] = "zwei" # change dictionary (and view object)
4 for i in v:
5     print("key:",i[0]+", value:",i[1])
```

Die Ausgabe ist
```
key: one, value: eins
key: two, value: zwei
```

Das Modul collections stellt Abwandlungen des Datentyps dict bereit (ChainMap, Counter, defaultdict, OrderedDict). Das Modul muss vor der Verwendung der Datentypen importiert werden, wie folgendes Beispiel zeigt.

```
1 # Beispiel zum Datentyp Counter
2 import collections
3 collections.Counter([1,2,3,2])
```

Mengen

- set (veränderliche Menge), z.B. `M = {"x", 8}`
- frozenset (unveränderliche Menge), z.B. `fs = frozenset({2, 3, "y"})`

Eine Menge ist eine Sammlung beliebiger Objekten, die jedes Objekt höchstens einmal enthält. Menge können, müssen aber nicht geordnet sein. Geordnet heißt: es gibt eine Ordnungsrelation für alle Elemente. Ein set mit der Schreibweise { ... } darf nicht mit einem dictionary verwechselt werden. `M = set()` erzeugt eine leere Menge.

Tabelle 8.3 führt Methoden und Operatoren für sets und frozensets auf und Tabelle 8.4 weitere Methoden für sets.

`x in M`	`True` wenn `x` $\in$ `M`, sonst False	`2 in {1, 4, 9}` $\rightarrow$ `False`
`x not in M`	`True` wenn `x` $\notin$ `M`, sonst False	`2 not in {1,4,9}` $\rightarrow$ `True`
`len(M)`	Anzahl der Elemente von `M`	`len( {1, 4, 9} )` $\rightarrow$ `3`
`min(M)`	kleinstes Element (mit Ordnung)	`M={1,2,3} ; min(M)` $\rightarrow$ `1`
`max(M)`	größtes Element (mit Ordnung)	`max{"a", "b", "c"}` $\rightarrow$ `'c'`
`M & m`	Durchschnitt von M und m	`{1,2,3} & {3,4}` $\rightarrow$ `{3}`
`M - m`	Differenz der Mengen M und m	`{1,2,3} - {3,4}` $\rightarrow$ `{1, 2}`
`M | m`	Vereinigung der Mengen M und m	`{2,3}|{3,4}` $\rightarrow$ `{2, 3, 4}`
`m >= M`	`True`, wenn `m` $\subseteq$ `M`, sonst `False`	`{3,4}.issuperset{3}` $\rightarrow$ `True`
`m <= M`	`True`, wenn `m` $\subseteq$ `M`, sonst `False`	`{3,4}.issubset{3}` $\rightarrow$ `True`

Tabelle 8.3: Methoden und Operatoren für sets und frozensets. `min` und `max` setzen eine Ordnungsrelation für alle Elemente voraus. Die Schreibweisen `m <= M` und `m.issubset(M)` sind gleichwertig, ebenso `m >= M` und `m.issuperset(M)`.

8.1.4 Ein- und Ausgaben, Dateien

Überschreiben der Ausgabezeile

Normalerweise schreibt der `print`-Befehl in eine neue Zeile. Um in der letzten Zeile eine laufend aktualisierte Information, zum Beispiel die Uhrzeit, ohne Zeilenvorschub

`M.add(x)`	Hinzufügen von `x` zu `M`	`M={6};M.add("a")` → `{6, 'a'}`
`M.clear()`	Entfernung aller Elemente	`M={6, 'a'}; M.clear()` → `{}`
`M.discard(x)`	Entfernung von `x`, falls es `x` gibt	`M={8} ; M.discard(9)` → `{8}`
`x = M.pop()`	entnimmt irgendein Element	`M={3,4,5}; M.pop()` → `{3, 5}`
`M.remove(x)`	Entfernung von `x`, sonst Fehler	`M={8,9} ; M.remove(9)` → `{8}`

Tabelle 8.4: Methoden für sets, nicht für frozensets (zusätzlich zu den in Tabelle xxx genannten). `pop` gibt das entnommene Element zurück (Fehler bei {}).

anzuzeigen, kann man wie in folgendem Beispiel vorgehen.

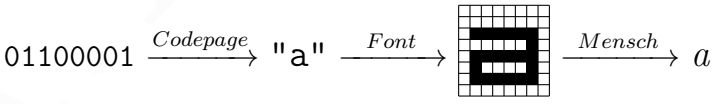
```
1 # Programm "ueberschreib" mit Python 3, AHG (2020)
2 import time
3 print(time.strftime("%H Uhr %M und %S Sekunden"),\
4 end="\r",flush=True)
```

Text- und Binärdateien

Python unterscheidet Textdateien (englisch: text files) und Binärdateien (binary files). In allen Dateien sind die Daten als Abfolge von Bits abgelegt:

Textdateien sind Dateien, deren Inhalt als Zeichenkette behandelt wird. Diese Art der Behandlung heißt *text mode*, im Gegensatz zum *binary mode*. Der wesentliche Unterschied zwischen Textdateien und Binärdateien beruht also auf der Interpretation durch Anwendungsprogramme.

Ein Vorteil von Textdateien ist, dass sie für Menschen mit einem Texteditor lesbar sind. Die Abbildung von Bitfolgen auf Zeichen heißt Codepage (Zeichensatztabelle). Python 3 benutzt in der Regel UTF-8 für alle Zeichen, die im Unicode enthalten sind.



Obige Zeichnung zeigt die Abbildung einer Bitfolge `01100001` auf ein Zeichen `"a"` (im Unicode durch eine Zahl benannt), dann auf eine digitalisierte Glyphe  (Gestalt des Zeichens) und schließlich die Wahrnehmung als Bedeutungsträger a (durch Menschen).

Auf Dateien greift man über ein Dateiobjekt zu. Das kann durch die Funktion `open()` erzeugt werden. Ihre wichtigsten Argumente sind eine Pfadangabe zur Datei und der Modus des Zugriffs, siehe Tabelle 8.5. Methoden zum Arbeiten mit Dateiobjekten zeigt Tabelle 8.6.

`"r"`	öffne Textdatei zum Lesen (englisch **r**ead), Standard (auch ohne Modus-Angabe)
`"w"`	öffne Textdatei zum Schreiben (englisch **w**rite), überschreibt bestehende Datei
`"a"`	öffne Textdatei zu ergänzendem Schreiben (englisch **a**ppend), überschreibt nicht
`"r+"`	öffne Textdatei zum Lesen und Schreiben, die Datei muss bereits bestehen

Tabelle 8.5: Wichtige Modi für den Zugriff auf Textdateien, Der Positionszeiger für Lesen und Schreiben wird bei `"r"`, `"w"` und `"r+"` auf den Anfang der Datei gesetzt, bei `"a"` ans Ende. Er wird beim Lesen und Schreiben weiterbewegt.

write(s)	schreibe Zeichenkette s in die Datei (`ergänze \n` für Zeilenende)
flush()	schließe Schreiben ab (übergebe Daten jetzt ans Betriebssystem)
read(n)	lies n Bytes der Datei in eine Zeichenkette, ohne n: ganze Datei
readline(n)	lies n Bytes einer Zeile der Datei, ohne n: ganze Zeile (einschl. `\n`)
seek(0)	setze den Zeiger für Lesen und Schreiben auf den Anfang der Datei

Tabelle 8.6: Wichtige Methoden für den Zugriff auf Textdateien

Ein offenes Dateiobjekt wird mit `close()` geschlossen. In folgendem Beispiel wird in eine Datei `datei.txt` geschrieben und die Aktion abgeschlossen.

```
1 #!/usr/bin/python3
2 f = open("datei.txt","w")
3 f.write("Quod scripsi, scripsi.")
4 f.close()
```

Manche Befehle verlangen Vor- und Nachbereitung, werden also in einem bestimmten Umfeld (Kontext) ausgeführt. Beispiel: Das Schreiben in eine Datei erfordert als Vorbereitung das Öffnen und als Nachbereitung das Schließen der Datei. Ist für einen Befehl ein sogenannter Kontextmanager definiert, übernimmt dieser die Vor- und Nachbereitung. Ein Kontextmanager wird mit dem with-Befehl erzeugt.

In folgendem Beispiel verwaltet der Kontextmanager ein Dateiobjekt `f` und stellt sicher, dass die Datei `datei.txt` nach dem Schreiben, auch wenn ein Fehler auftritt, schließlich geschlossen wird.

```
1 #!/usr/bin/python3
2 # schreibe Zeichenkette (String) in eine Textdatei
3 with open("datei.txt", "w") as f:
4     f.write("Dies ist ein sinnvoller Satz.")
```

In einem Programm kann man Daten auf verschiedene Weise in eine Textdatei schreiben oder aus einer Datei lesen. Folgendes Programm zeigt das Anhängen von Text und das Lesen einer Textdatei. Dabei wird das Dateiobjekt `f` als Kontextmanager erzeugt.

```
# Programm "buschzitat" mit Python 3, AHG (2020)
import os

L=[] # leere Liste (append ergänzt am Ende)
L.append("Stets findet Überraschung statt,")
L.append("da, wo man's nicht erwartet hat.")

for x in L:
    with open (datei, "a") as f :
        f.write (x+"\n") # mit Zeilenvorschub
# Öffnen der Datei, "a" (append) erzeugt sie bei Bedarf

with open (datei, "r") as f :
    t=f.read () # Typ str (Zeichenkette)
print(t)
```

Zufällige Datei-Auswahl

Nach Laden der Pythonmodule `glob` und `random` kann man in einem Verzeichnis eine Datei zufällig auswählen. Im folgendem Programm wird eine Datei des Arbeitsverzeichnisses (durch `.` angegeben) mit der Endung `.mp3` gewählt.

```
# Programm "zufallsdatei" mit Python 3, AHG (2020)
import glob, random
datei = random.choice(glob.glob("./*.mp3"))
```

Textspeicherung aus WebVTT-Untertiteldatei

Folgendes Programm schreibt die Untertitel eines Videos, die in einer WebVTT-Datei (`x.vtt`) gespeichert sind (siehe Seite 203), in eine normale Textdatei (`y.txt`).

```
# Programm "vtt2txt" mit Python 3, AHG (2020)
from itertools import islice
with open('x.vtt') as ad, open('y.txt', 'w') as nd:
    for l in islice(ad, 3, None):
        if (l != "\n") and not ("-->" in l): nd.write(l)
```

Die Funktion `islice` entfernt Elemente am Anfang (und möglicherweise auch am Ende) eines Iterators. Im Beispiel werden die ersten drei Zeilen einer Datei verworfen.

8.1.5 Kontrollstrukturen

Befehle werden der Reihe nach ausgeführt, außer in Kontrollstrukturen. Diese erlauben es, Befehle abhängig vom Wert einer Variablen (oder eines Ausdrucks) auszuführen oder nicht. Damit sind Verzweigungen des Programmablaufs möglich. Außerdem können Fehler abgefangen und in einem besonderen Programmteil verarbeitet werden. Wiederholungen von Befehlsblöcken, abhängig vom Wert einer Variablen, sind auch möglich.

- Bedingte Anweisungen (conditionals)
  - ohne Verzweigung: if
  - mit Verzweigung: if (– elif) – else
- Schleifen (loops)
  - while-Schleife
  - for-in-Schleife
- Ausnahmebehandlung (exception handling)

if

Nach dem Schlüsselwort `if` steht eine Variable oder ein Ausdruck, aus dem (mit Hilfe der Funktion bool(), siehe Seite 283) ein boolescher Wert (`True` oder `False`) gebildet wird. Anschließend steht ein Doppelpunkt. Es folgt ein Befehlsblock, der durch Einrückung der Zeilen gekennzeichnet ist. Die Befehle des Blocks werden genau dann ausgeführt, wenn der Ausdruck `True` ergab. Im folgenden Programm hat `a != 0` (a ungleich 0) den Wert `False` und der Befehlsblock wird nicht ausgeführt.

```
1 # Beispiel zur üblichen if-Anweisung
2 a = 0           # Wertzuweisung an Variable a
3 if a!=0:        # falls Bedingung wahr:
4     b = 1/a     # erster Befehl des Blocks
5     print(b)    # zweiter Befehl des Blocks
```

Besteht der bedingte Befehlsblock aus nur einer Zeile, kann der Befehl, mit oder ohne Leerzeichen, nach dem Doppelpunkt angefügt werden. Im folgenden Beispiel wird `Hallo` ausgegeben. `end=""` entfernt das Zeilenende-Zeichen.

```
1 # Beispiel zur einzeiligen if-Anweisung
2 if True:print("Ha",end="")
3 if not False: print"llo!")
```

if – else

Mit `if - else` gibt es zwei alternative Befehlsblöcke. Ergibt der Ausdruck nach dem `if` den Wert `True`, dann wird der erste Block ausgeführt, bei `False` der zweite. Im folgenden Beispiel ist der Ausdruck `True` und es wird die Zeichenkette `zu wenig` ausgegeben.

```
# Beispiel zur üblichen if-else-Anweisung
a, b = 0, 8                     # a ist 0 und b ist 8
if a < b:                       # falls Bedingung wahr:
    print("zu wenig")           # Befehlsblock für True
else:                           # andernfalls:
    print("nicht zu wenig")     # Befehlsblock für False
```

Bestehen alternative Befehlsblöcke jeweils nur aus einem kurzen Befehl, kann man die ganze `if-else`-Anweisung in einer Zeile schreiben. Ergibt der Ausdruck nach dem `if` den Wert `True`, wird der Befehl vor dem `if` ausgeführt, sonst der nach dem `else`. Folgendes gibt `Nein` aus.

```
# Beispiel zur einzeiligen if-else-Anweisung
a, b = 2, 1                          # a ist 2 und b ist 1
print("Ja") if a < b else print("Nein")   # Alternative
```

if – elif – else

Die `if-elif-else`-Anweisung ist eine Erweiterung der `if-else`-Anweisung. Hat der Ausdruck nach dem `if` den Wert `True`, wird der Befehlsblock nach dem `if` (erster Block) ausgeführt. Bei `False` wird der Ausdruck nach dem `elif` ausgewertet. Ist dieser `True`, wird der Befehlsblock nach dem `elif` (zweiter Block) ausgeführt. Andernfalls wird der Befehlsblock nach dem `else` (dritter Block) ausgeführt. Folgendes gibt die Zeichenkette `zu viel` aus.

```
# Beispiel zur if-elif-else-Anweisung
a, b = 2, 1    # a ist 2 und b ist 1
if a < b:      # falls 1. Bedingung wahr:
    print("zu wenig")
elif a > b:    # sonst, falls 2. Bedingung wahr:
    print("zu viel")
else:          # andernfalls:
    print("gerade richtig")
```

Man kann statt eines `elif`-Block mehrere `elif`-Blöcke setzen und so mehr Fälle überprüfen. Der `else`-Block darf auch fehlen. Das Programm wird dann nach dem letzten `elif`-Block fortgesetzt, unabhängig davon, ob der `if`-Block oder der `elif`-Block ausgeführt wurden oder keiner von beiden.

Auswahl aus vielen Möglichkeiten

Wenn viele Fälle geprüft werden sollen, ist eine `if-elif-else`-Anweisung mit mehreren `elif`-Blöcken keine schöne Lösung. Eine switch/case Kontrollstruktur, wie in der

Programmiersprache C, gibt es in Python nicht. Man kann das gleiche jedoch mit einem dictionary erreichen, dessen Werte Funktionen sind, wie folgendes Beispiel zeigt.

```
1  # Program case.py makes a case
2  def f_a():
3      print("Decision: case a")
4  def f_b():
5      print("Decision: case b")
6  def f_c():
7      print("Neither a nor b")
8  dic = {
9  "a": f_a,
10 "b": f_b,
11 }
12
13 print("a for function f_a")
14 print("b for function f_b")
15 x = input("Enter a or b: ")
16 dic.get(x,f_c)()
```

Das Programm bittet den Nutzer in Zeile 15 um eine Texteingabe über die Tastatur. Nach Eingabe von `a` wird die Funktion `f_a` ausgeführt. Diese gibt auf dem Bildschirm die Zeichenkette `Decision: case a` aus. Nach Eingabe von `b` wird die Funktion `f_b` ausgeführt und `Decision: case b` ausgegeben. Andere Eingaben sind kein Schlüssel (key) im dictionary `dic` und dann wird die als Standardwert genannte Funktion `f_c` ausgeführt und `Neither a nor b` auf dem Bildschirm ausgegeben.

while

Der Ausdruck nach dem Schlüsselwort `while` wird (mit Hilfe der Funktion bool(), siehe Seite 283) zu einem booleschen Wert ausgewertet. Bei `True` wird der folgende Befehlsblock ausgeführt. Bei `False` wird das Programm nach diesem Block fortgesetzt. Das folgende Programm gibt innerhalb der `while`-Schleife nacheinander die Werte 3, 2 und 1 aus. Danach wird `Null erreicht!` ausgegeben.

```
1  # Beispiel zur while-Schleife
2  n = 3
3  while n > 0:                    # Beginn der Schleife
4      print("Zahl:", n)           # (im Befehlsblock)
5      n = n - 1                   # (im Befehlsblock)
6  print("Null erreicht!")
```

for – in

In der ersten Zeile steht das Schlüsselwort `for`, dann der Name einer Variablen, die nur innerhalb der `for`-Schleife gebraucht wird. Darauf folgen das Schlüsselwort `in`, ein iterierbares Objekt und ein Semikolon. Iterierbare Objekte sind Instanzen von Datentypen mit mehreren Elementen (Liste, Menge, Dictionary, ...). Die Variable referenziert nacheinander alle Elemente des iterierbaren Objekts und führt jedes Mal den Befehlsblock aus, welcher der ersten Zeile folgt. Folgendes Programm gibt nacheinander alle Elemente einer Liste aus.

```
1 # Beispiel zur for-in-Schleife
2 namen = ["Peter", "Paul", "Mary"]
3 for i in namen:                # Beginn der Schleife
4     print(i)                   # (im Befehlsblock)
```

Ausnahmebehandlung: exceptions

Es gibt zwei Arten von Fehlern in einem Programm: syntaktische und semantische. Bei einem Fehler erzeugt Python eine Instanz vom Datentyp einer bestimmten exception (Ausnahme). Man sagt: es wird eine Ausnahme „geworfen“ (englisch: to throw an exception). Syntaxfehler (syntax errors) entstehen zum Beispiel durch falsche Schreibweise von Schlüsselwörtern und verhindern die Ausführung des Programms, indem eine exception vom Typ `SyntaxError` geworfen wird. Semantische Fehler treten während der Ausführung des Programms auf, wenn zum Beispiel versucht wird, durch 0 zu dividieren.

Die exception (Ausnahme) kann im Programm gefangen werden. Das heißt, dass nach dem Auftreten der exception ein dafür geschriebener Anweisungsblock ausgeführt wird. Die Behandlung einer exception nennt man exception handling.

Tut man das nicht, wird das Programm mit einer Fehlermeldung abgebrochen, in welcher der Ursprung des Fehlers zurückverfolgt wird (traceback).

Der Abschnitt des Quellcodes, in dem exceptions gefangen werden sollen, wird in einem Befehlsblock nach `try:` geschrieben. Danach kann man Befehlsblöcke schreiben, die jeweils einen bestimmten Typ von exception behandeln. Ein einfaches Beispiel:

```
1 # Abfangen einer Division durch 0
2 try:
3     x = int(input("Gib eine Zahl ein: "))
4     print(str(100/x))
5 except ZeroDivisionError:
6     print("Division durch 0 geht nicht!")
7 print("Ende, auch nach Eingabe von 0")
```

In obigem Beispiel wird eine exception vom Typ `ZeroDivisionError` geworfen, wenn nach der Eingabe von 0 (Zeile 3) versucht wird, durch Null zu teilen (Zeile 4). Diese

exception wird in Zeile 5 (`except ZeroDivisionError:`) gefangen und der Befehl in Zeile 6 wird ausgeführt. Danach wird das Programm in Zeile 7 fortgesetzt.

Alle anderen Typen von exceptions werden nicht gefangen, sondern führen zum Abbruch des Programms mit einer entsprechenden Fehlermeldung. Tritt kein Fehler auf, wird keine exception geworfen und das Programm wird nach Abarbeitung des Blocks aus Zeilen 3 und 4 in Zeile 7 fortgesetzt.

Im folgenden Beispiel werden verschiedene exceptions behandelt.

```
1  # Abfangen verschiedener exceptions
2  try:
3      x = int(input("Gib eine Zahl ein: "))
4      print(str(100/x))
5  except ZeroDivisionError:
6      print("Division durch 0 geht nicht!")
7  except KeyboardInterrupt:
8      print("\nAbbruch durch Nutzer (Ctrl-C)!")
9  except:
10     print("Ein anderer Fehler ist aufgetreten!")
11 print("Ende, mit oder ohne einen Fehler.")
```

Nach der Eingabe von 0 wird eine `ZeroDivisionError` exception geworfen. Diese wird, wie im vorigen Beispielprogramm, gefangen. Falls der Nutzer, während das Programm auf eine Eingabe wartet, die Ausführung mit der Tastenkombination `Ctrl-C` abbricht (siehe Seite 12), wird eine `KeyboardInterrupt:` exception geworfen. Diese wird in Zeilen 7 und 8 gefangen und behandelt.

Alle anderen Fehler führen in einer sonstigen exception (default exception), die in Zeilen 9 und 10 gefangen und behandelt wird. Das Abfangen von sonstigen exceptions muss nach dem Abfangen der bestimmten exceptions stehen. In obigem Beispiel tritt ein `ValueError` auf, wenn eine leere Eingabe gemacht wird (nur die ENTER-Taste gedrückt wird). Dieser Fehler wird als sonstige exception gefangen.

Unabhängig davon ob im `try`-Block (in Zeilen 3 und 4) ein Fehler auftrat (eine exception geworden wurde) oder nicht, wird das Programm in Zeile 11 fortgesetzt.

Zusicherung: assert

Um bei der Programmentwicklung sicherzustellen, dass ein Programm immer das tut, was wir wollen, können wir an wichtigen Stellen prüfen, ob bestimmte Bedingungen erfüllt werden. Dafür gibt es den `assert` Befehl. Dem Schlüsselwort `assert` folgt die Bedingung, die erfüllt sein soll. Danach kann, nach einem Komma, eine Zeichenkette stehen, welche im Fehlerfall angezeigt wird. Beispiel:

```
1  x=42 ; x = x + 1
2  assert x==42, "Variable x ungleich 42"
```

In obigem Beispiel wird das Programm mit einem Fehler beendet und die Meldung `AssertionError: Variable x ungleich 42` angezeigt. Wenn die Bedingung in einem `assert` Befehl erfüllt ist, wird das Programm normal (ohne Meldung) fortgesetzt.

8.1.6 Operatoren und Funktionen, Mathematik

Operatoren kann man benutzen, um aus Instanzen eines Datentyps eine neue Instanz zu bilden. Arithmetische Operatoren (zum Beispiel +, –, ∗, /, ∗∗, // und %) arbeiten mit numerischen Datentypen. Zum Beispiel ist `2**3` $= 2^3 = 8$. `a // b` bildet die nächstkleinere Ganzzahl des Quotienten a / b und `a % b` den zugehörigen Rest der ganzzahligen Division. Beispiel: `7 // 3` ergibt 2 und `7 % 3` ergibt 1 .

Für Zeichenketten und andere sequenzielle Datentypen gibt es den Operator `+` für die Konkatenation (siehe unten). Das Pluszeichen bedeutet also entweder Addition (bei numerischen Datentypen) oder Konkatenation (bei sequenziellen Datentypen).

Vergleichsoperatoren (`==`, `!=`, `<`, `<=`, `>`, `>=`) liefern `True` oder `False`, je nachdem ob ein Vergleich zutrifft oder nicht. Beispiel: `1 != 2` ergibt `True`, da $1 \neq 2$ ist. Vergleichsergebnisse werden oft mithilfe logischer Operatoren (`not`, `and`, `or`) verknüpft; Beispiel: `(1 == 2) or (1 != 2)` ergibt `True`.

Anders als in der Programmiersprache C stellt das Zuweisungssymbol `=` in Python keinen Operator (operator) dar, sondern eine Anweisung (statement), genauer eine Zuweisungsanweisung (assignment statement).

Operatorrangfolge (operator precedence, order of operations)

Wenn ein Ausdruck mehrere Operatoren enthält, aber keine Klammern, die Operatoren in der Reihenfolge ihres Rangs (precedence) ausgewertet.[3] Operatoren gleichen Rangs werden von links nach rechts ausgewertet. Tabelle 8.7 zeigt die Rangfolge wichtiger Operatoren. Geklammerte Ausdrücke werden vor ungeklammerten ausgewertet.

Arithmetische Operatoren

```
2 * 3          ergibt  6    (Datentyp int)
2 ** 3         ergibt  8    (Datentyp int)
1 / 2          ergibt  0.5  (Datentyp float)
7 // 2         ergibt  3    (Datentyp int, nächstkleinere Ganzzahl)
-6.9 // 2.1    ergibt  -4.0 (nächstkleinere Ganzzahl → float)
7 % 2          ergibt  1    (Rest von 7 // 2 , nur sinnvoll mit Typ int)
```

[3] Eine Ausnahme ist zum Beispiel der Vorzeichenoperator im Exponenten: $2 * * -1$ ergibt 0,5.

Operator	Bedeutung
x ** y	Potenzierung (exponentiation)
$+x$, -x, $\sim x$	Vorzeichen, bitweises NICHT (bitwise NOT)
$x * y$, x / y, $x // y$, $x \% y$	Multiplikation, Division, ganzzahlige Division, Rest
$x + y$, x - y	Addition und Subtraktion
$x << y$, $x >> y$	bitweise Verschiebungen (shifts)
$x \& y$	bitweises UND (bitwise AND)
$x \hat{} y$	bitweises ausschließendes ODER (bitwise XOR)
$x \mid y$	bitweises nicht ausschließendes ODER (bitwise OR)
$x<y$, ..., x `in` y, ...	Vergleichsoperatoren und Tests (comparisons and tests)
`not` x	logisches NICHT (boolean NOT)
x `and` y	logisches UND (boolean AND)
x `or` y	logisches ODER (boolean OR)

Tabelle 8.7: Nach absteigendem Rang geordnete Operatoren (ohne Klammerungen)

Numerische Vergleichsoperatoren

`a == b`	`True` genau dann, wenn `a` und `b` den gleichen Wert haben
`a != b`	`False` genau dann, wenn `a` und `b` den gleichen Wert haben
`a < b`	`True` genau dann, wenn `a` einen kleineren Wert als `b` hat
`a <= b`	`False` genau dann, wenn `a` einen größeren Wert als `b` hat
`a > b`	`True` genau dann, wenn `a` einen größeren Wert als `b` hat
`a >= b`	`False` genau dann, wenn `a` einen größeren Wert als `b` hat

Die letzten vier Operatoren sind nicht für komplexe Zahlen definiert, da der Körper der komplexen Zahlen nicht geordnet ist.

Vergleichsoperatoren für Zeichenketten (Strings)

`a == b`	`True` genau dann, wenn `a` und `b` den gleichen Wert haben
`a != b`	`False` genau dann, wenn `a` und `b` den gleichen Wert haben

Es gibt weitere Operatoren, welche mit der lexikographischen Ordnung vergleichen.

Logische Operatoren

Logische Operatoren dienen der Verknüpfung der logischen Werte `True` und `False` (Datentyp `bool`), oder von Ausdrücken, die zu logischen Werten ausgewertet werden. Das Ergebnis ist wieder ein logischer Wert.

Es gibt Operatoren für die unäre Verknüpfung `not` (ein Operand) und für die binären Verknüpfungen `and` und `or` (zwei Operanden). Diese Verknüpfungen werden durch Wahrheitstafeln beschrieben, siehe Tabelle 8.8.

x	not x
True	False
False	True

x	y	x and y
True	True	True
True	False	False
False	True	False
False	False	False

x	y	x or y
True	True	True
True	False	True
False	True	True
False	False	False

Tabelle 8.8: Wahrheitstafeln der logischen Operatoren `not`, `and` und `or`

In einem Ausdruck mit mehreren logischen Operatoren, der keine runden Klammern enthält, wird die Reihenfolge der Anwendung durch die Rangfolge der Operatoren bestimmt. Diese ist (zuerst → zuletzt): `not`, `and`, `or`. Es ist in solchen Fällen aber besser, die Reihenfolge der Anwendung durch Klammerung deutlich festzulegen. Die in runden Klammern gesetzten Ausdrücke werden zuerst ausgewertet. Bei geschachtelten Klammerungen wird die innerste Klammer zuerst ausgewertet.

`True or True and False` ergibt `True`

`(True or True) and False` ergibt `False`

`True or (True and False)` ergibt `True`

Vergleichsoperatoren haben einen höheren Rang als logische Operatoren, werden also vorher ausgewertet.

`1 < 2 or 1 > 2` ist gleich `True or False` und ergibt `True`

`1 < 2 and 1 > 2` ist gleich `True and False` und ergibt `False`

Funktionen

Eine Funktion (function) ist ein Block (Folge von Befehlen), der mit einem Namen versehen wird (Definition) und beliebig oft zur Ausführung gebracht werden kann (Aufruf). Eine Funktion kann Daten entgegennehmen, welche in der Definition der Funktion durch Parameter repräsentiert werden.

Literale oder Referenzen, die beim Aufruf der Funktion für die Parameter übergeben werden, heißen Argumente. Jede Funktion gibt eine Instanz zurück (Rückgabewert). Der Rückgabewert kann in der Funktionsdefinition durch einen oder mehrere return-Befehle festgelegt werden. Ansonsten ist der Rückgabwert `None`.

Eine Funktionsdefinition besteht aus dem Schlüsselwort `def`, dem Namen, einem runden Klammerpaar und einem Doppelpunkt, gefolgt vom Befehlsblock. Die runden Klammen können beliebig viele Parameter enthalten, möglicherweise keine.

Parameter können Pflichtparameter sein, die beim Aufruf übergeben werden müssen, oder optionale Parameter, die übergeben werden können. Optionale Parameter müssen in der Definition mit einem Standardwert (default value) belegt werden, der verwendet wird, wenn kein anderer Wert übergeben wird. Optionale Parameter stehen nach den Pflichtparametern.

Im Befehlsblock sind die Parameter Referenzen auf Instanzen, deren Wert beim Aufruf festgelegt wurde. Ein Beispiel mit einem Pflichtparameter (`x`) und zwei optionalen Parametern (`a` und `b`):

```
1 def gerade(x, a=1, b=0):
2     return a*x+b
3 print( gerade(5) )
4 print( gerade(5, 2, 3) )
```

Die Ausgabe des Programms ist `5`, denn $5 \cdot 1 + 0 = 5$, und in der nächsten Zeile `13`, denn $5 \cdot 2 + 2 = 13$.

Die Zuordnung der übergebenen Werte (Argumente) zu den Parametern erfolgte durch die Position der Parameter innerhalb der runden Klammern der Funktionsdefinition. Diese Art von Argumenten heißt *positional arguments*. Man kann die Argumente in beliebiger Reihenfolge übergeben, wenn man dazu schreibt, zu welchem Parameter sie gehören.

```
1 def gerade(x, a=1, b=0):
2     return a*x+b
3 print( gerade(a=2, b=0, x=5) )
```

Die Ausgabe ist wieder `13`. Diese Art von Argumenten heißt *keyword arguments*. Man kann positional arguments gleichzeitig verwenden, aber erst die positional arguments und danach keyword arguments, die noch nicht als positional arguments übergeben wurden.

```
1 def gerade(x, a=1, b=0):
2     return a*x+b
3 print( gerade(6, b=2) )
```

Die Ausgabe ist `8`, denn $6 \cdot 1 + 2 = 8$.

Namen, die außerhalb von Funktionen vergeben werden, gehören zum *globalen Namensraum* eines Programms. Innerhalb dieses Namensraums müssen Namen eindeutig sein. Jede Funktion hat einen eigenen Namensraum, den *lokalen Namensraum* der Funktion. Die Parameter und andere Referenzen, die in der Funktionsdefinition erzeugt werden, sind nur im Block der Funktion gültig.

Im Funktionsblock können globale Referenzen gelesen werden, aber normalerweise nicht geändert (geschrieben) werden. Um eine globale Referenz verändern zu können, muss im Funktionsblock ein `global`-Befehl für diese Referenz stehen. Beispiel:

```
1 a = b = 1
2 def dummy():
3     global b
4     a = 2
```

```
5       b = 2
6   dummy()
7   print(a, b)
```

Die Ausgabe ist `1 2`. Die lokale Referenz `a` der Funktion `dummy` gilt nur im lokalen Namensraum und hat nicht mit der globalen Referenz `a` zu tun. Dass sie den gleichen Namen haben, ist unschädlich, weil sie in unterschiedlichen Namensräumen gekapselt sind. Die globale Referenz `b` wurde in der Funktion `dummy` verändert, weil `b` mit dem `global`-Befehl in den lokalen Namensraum übernommen wurde.

Built-in functions

Die Funktion `abs()` bildet den Absolutbetrag einer Ganzzahl oder Gleitkommazahl. Für komplexe Zahlen wird der Betrag als Gleitkommazahl zurückgegeben. Bei booleschen Werten wird für `True` die Ganzzahl 1 und für `False` die Ganzzahl 0 zurückgegeben. Folgendes Programm gibt die Werte `42`, `1.8`, `1` und `5.0` aus.

```
1   # Beispiel zur Funktion abs
2   print(abs(42))        # int (Ganzzahl)
3   print(abs(-1.8)))     # float (Gleitkommazahl)
4   print(abs(True))      # bool (logischer Wert)
5   print(abs(4+3j))      # complex (komplexe Zahl)
```

Die Funktion bool() wandelt ihr Argument in einen logischen Wert (`True` oder `False`) um. Dies ist die Grundlage für Entscheidungen in Kontrollstrukturen (siehe oben). Ist das Argument ein Ausdruck, wird er ausgewertet. Schließlich erhält man eine Instanz. Ist diese nicht vom Typ bool, wird ein boolescher Wert erzeugt, siehe Tabelle 8.9.

Datentyp	Wert	bool()
NoneType	None	`False`
int	0	`False`
int	ungleich 0	`True`
float	0.0	`False`
float	ungleich 0.0	`True`
bool	`False`	`False`
bool	`True`	`True`
complex	0+0j	`False`
complex	ungleich 0+0j	`True`
str	""	`False`
str	ungleich ""	`True`

Tabelle 8.9: Auswertung verschiedener Datentypen durch die Funktion `bool()`

Alle sequenziellen Datentypen (str, list, tuple), Zuordnungen (dict) und Mengen (set, frozenset) ergeben als Argument von bool() genau dann `False`, wenn sie kein Element enthalten. Ist `x` eine Zeichenkette (str), kann mit `bool(x)` geprüft werden, ob `x` leer (`""`) ist oder nicht. Ähnliche Prüfungen werden oft in Kontrollstrukturen verwandt.

Die Funktion `max()` gibt das Maximum der Elemente eines iterierbaren Objekts zurück. Folgendes gibt `85` und `5.3` aus.

```
# Beispiel zur Funktion max
x = {38,26,85,27,3}    # Menge (set)
y = [2.0, 5.3, 3.9]    # Liste (list)
print (max(x))
print (max(y))
```

Die Funktion `min()` gibt das Minimum der Elemente eines iterierbaren Objekts zurück. Folgendes gibt `3` und `2.0` aus.

```
# Beispiel zur Funktion min
x = {38,26,85,27,3}    # Menge (set)
y = [2.0, 5.3, 3.9]    # Liste (list)
print (min(x))
print (min(y))
```

Die Funktion `round()` rundet eine Gleitkommazahl (erstes Argument) auf eine vorgegebene Zahl von Nachkommastellen (zweites Argument). Folgendes gibt `3.14` aus.

```
# Beispiel zur Funktion round
x = 3.1415926)        # Gleitkommazahl
print(round(x,2))     # Rundung auf zwei Nachkommastellen
```

Rekursive Funktionen

Eine Funktion kann sich selbst aufrufen. Eine solche nennt man rekursive Funktion. In folgendem Beispiel wird die Fakultät einer natürlichen Zahl berechnet. Die Funktion wird mit `3` als Argument aufgerufen. Als Ergebnis wird $3! = 6$ ausgegeben.

```
# Berechnung der Fakultät als rekursive Funktion
def fak(n):
    if n == 0:
        return 1
    else:
        return n*fak(n-1)
print(fak(3))
```

Komplexe Zahlen

Als Literal für die imaginäre Einheit dient, wie in der Elektrotechnik, `j` oder `J`. Das Symbol wird unmittelbar nach dem Imaginärteil geschrieben. So kann es nicht zu einer Verwechslung mit einer Referenz kommen, die mit j (oder J) beginnt.

Instanzen vom Typ `complex` haben die Eigenschaften (Attribute) `real` und `imag`, welche den Realteil beziehungsweise den Imaginärteil enthalten. Außerdem gibt es die Methode `conjugate`, mit welcher die konjugiert komplexe Zahl gebildet wird. Den Betrag $|x|$ einer komplexen Zahl x liefert der Funktion $abs(x)$.

```
1 # erstes Beispiel zu komplexen Zahlen
2 x = 1.3 - 2.4j
3 y = 3.5 + 4j
4 z = x + y
5 print("Realteil = ",z.real)
6 print("Imaginärteil = ",z.imag)
7 print("Konjugierte = ",z.conjugate())
8 print("Betrag = ",abs(z))
```

ergibt `Realteil = 4.8`
und `Imaginärteil = 1.6`
und `Konjugierte = (4.8-1.6j)`
und `Betrag = 5.059644256269407` in jeweils einer Zeile.

Weitere Funktionen für komplexe Zahlen werden vom Modul `cmath` bereitgestellt.

`cmath.phase()` gibt den Polarwinkel (das Argument) φ zurück.

Die Funktion `cmath.polar(x)` gibt die Polarkoordinaten $|x|$ und φ einer komplexen Zahl x als Paar (Datentyp `tuple`) zurück.

Die Funktion `cmath.rect(`$|x|$`, `φ`)` wandelt die Polarkoordinaten einer komplexen Zahl in die Normalform $\Re(x) + \Im(x)\,\mathrm{j}$ mit dem Datentyp `complex`.

```
1 # zweites Beispiel zu komplexen Zahlen
2 import cmath
3 x = -4 + 3j
4 print(cmath.polar(x))
```

ergibt `(5.0, 2.498091544796509)`

8.1.7 Zeichenketten und Reguläre Ausdrücke

Literale und Glyphen

Die direkte Darstellung (ohne Referenzierung) des Werts einer Instanz nennt man Literal. Für Zeichenketten oder strings, also Instanzen des Datentyps str, gibt es Zeichenketten-

Literale (englisch string literals). Ein Zeichenketten-Literal ist entweder ein *shortstring*, vor dem ein Präfix stehen kann (zum Beispiel `r`, siehe Seite 287), oder ein *longstring*. Vereinfachend behandeln wir hier keine longstrings.

Ein *shortstring* ist eine Zeichenkette, die entweder in einfachen Anführungszeichen (englisch: single quotes; Beispiel: `'Jena'`) oder in doppelten Anführungszeichen (double quotes) eingeschlossen ist (`"Jena"`).

Ein besonderes Zeichen ist der Zeilenumbruch, für den Python (ebenso wie Linux, aber anders als Microsoft Windows) das Kontrollzeichen (englisch: control character) *Line Feed* (ASCII LF) benutzt. In einem shortstring ist das Kontrollzeichen *Line Feed* verboten und wird durch `\n` ersetzt.

Die Gestalt eines geschriebenen oder gedruckten Zeichens heißt *Glyphe*. Abbildung 8.1 [4] zeigt verschiedene Glyphen des lateinischen Kleinbuchstabens a (Zeichen `U+0061` im Unicode Zeichensatz UTF-8). Sonderzeichen sind Elemente einer Zeichenkette, die nicht mit einer Glyphe abgebildet werden (Steuerzeichen, wie jenes, welches einen Zeilenvorschub bewirkt, siehe unten) oder auf der Tastatur nicht vorgegeben sind (zum Beispiel der griechische Buchstabe π).

Abbildung 8.1: Glyphen

Escape-Sequenzen

Um in string-Literalen auch Sonderzeichen wiederzugeben, kann man mit sogenannten Escape-Sequenzen arbeiten. Sie bestehen in der Regel aus dem Zeichen \ (Backslash) und einem oder mehreren weiteren Zeichen. Häufig wird \n für einen Zeilenvorschub (LF) verwendet. Tabelle 8.10 listet Escape-Sequenzen für Sonderzeichen auf.

LF	`\n`	ϵ	`\u03B5`	ω	`\u03c9`	$\geq$	`\u2265`
CR	`\r`	λ	`\u03bb`	Ω	`\u03a9`	$\cdot$	`\u00b7`
\	`\\`	Λ	`\u039b`	^{o}	`\u00b0`	$\div$	`\u00F7`
'	`\'`	μ	`\u03bc`	^{2}	`\u00b2`	$\pm$	`\u00b1`
"	`\"`	π	`\u03c0`	^{n}	`\u207F`	∞	`\u221e`
€	`\u20ac`	Π	`\u03a0`	$\surd$	`\u221a`	$\rightarrow$	`\u2192`
α	`\u03b1`	ρ	`\u03c1`	$^{1}/_{4}$	`\u00bc`	Å	`\u00C5`
β	`\u03b2`	σ	`\u03c3`	$^{1}/_{3}$	`\u2153`	$\sim$	`\u007E`
γ	`\u03b3`	Σ	`\u03a3`	$^{1}/_{2}$	`\u00bd`	{	`\u007B`
δ	`\u03b4`	φ	`\u03c6`	$\approx$	`\u2248`	|	`\u007C`
Δ	`\u0394`	Φ	`\u03a6`	$\leq$	`\u2264`	$*$	`\u002A`

Tabelle 8.10: Sonderzeichen können durch Escape-Sequenzen dargestellt werden.

Damit ergibt

[4] Die Abbildung wurde vom Urheber Wickey-nl auf Wikimedia Commons gemeinfrei gegeben.

```
print("R = 80 \u03a9\nI = 4 \u00b1 1 \u00b5A")
```

die Ausgabe

```
R = 80 Ω
I = 4 ± 1 µA
```

Wenn ein string-Literal das Zeichen \ (Backslash) enthält, dies aber nicht den Beginn einer Escape-Sequenz markieren soll, kann ein Backslash durch die Escape-Sequenz \\ statt durch \ bezeichnet werden. Damit ergibt

```
print("R = 80 \\u03a9")
```

die Ausgabe

```
R = 80 \u03a9
```

Es gibt noch eine andere Schreibweise für strings, bei denen \ nicht als Beginn einer Escape-Sequenz gedeutet wird. Dies ist, unter anderem, bei der Niederschrift von Regulären Ausdrücken sinnvoll, wo \ eine besondere Bedeutung bekommt (siehe unten). Diese alternative Schreibweise, englisch *raw string* genannt, unterscheidet sich vom gewöhnlichen string-Literal durch ein vorangestelltes `r` oder `R`. So ergibt

```
print(r"R = 80 \u03a9\nI = 4 \u00b1 1 \u03bcA")
```

die Ausgabe

```
R = 80 \u03a9\nI = 4 \u00b1 1 \u03bcA
```

weil mit dem raw string keine Escape-Sequenzen in Sonderzeichen gewandelt werden.

Methoden für strings (Zeichenketten)

Der Datentyp `str` der strings ist ein sequenzieller Datentyp. Die in Tabelle 8.1 auf Seite 267 aufgeführten Operatoren und Methoden sind anwendbar. Beispiele:

```
"Je" + "na"    ergibt  "Jena"   (Konkatenation)
"u" in "Jena" ergibt   False    ("u" ist nicht in "Jena" enthalten)
```

Nicht für alle sequenziellen Datentypen, aber für strings, gibt es die in Tabelle 8.11 aufgeführten Methoden.

Whitespaces sind Leerzeichen, Tabulatoren und Zeilenvorschübe: " ", \n, \v, \t, \f, \r. Eine Konstante, die in einer Zeichenkette alle whitespaces enthält, wird im Modul `string` definiert (siehe Seite 293).

Gegeben sei ein Text, der aus mehreren Zeilen besteht, die durch Zeilenumbrüche `\n` getrennt sind. Das folgende Beispiel (mit nur zwei Zeilen) zeigt, wie man die erste Zeile (einschließlich des Zeilenumbruchs an ihrem Ende) entfernen kann.

`A.find(x)`	Index von `x` in `A`; `-1`, falls $\notin$ `A`	`"Jena".find("a")` → `3`
`A.replace(x,y)`	in `A` Ersetzung aller `x` durch `y`	"oxo".replace("o","i") → "ixi"
`A.strip()`	ohne Whitespaces vorn, hinten	`" abc\n".strip()` → `"abc"`
`A.split()`	von Whitespaces getrennte Teile	`" a b".split()` → `["a","b"]`
`A.lower()`	Groß- → Kleinbuchstaben	`"AaBb".lower()` → `["aabb"]`
`A.upper()`	Klein- → Großbuchstaben	`"AaBb".upper()` → `["AABB"]`
`A.join(B)`	Verbindung der Elemente von `B`	`"x".join(["a","b"])` → `"axb"`

Tabelle 8.11: Methoden für Zeichenketten (Strings)

```
1 fasta = "> comment\nMALWMRLLPLLALLALWGPDPAAAFVNQH"
2 raw = "\n".join(fasta.split("\n")[1:]); print(raw)
```

ergibt `MALWMRLLPLLALLALWGPDPAAAFVNQH`

Mit `fasta.split("\n")` wird aus dem Text `fasta` eine Liste der Zeilen ohne Zeilenumbrüche `\n`. `[1:]` löscht aus dieser Liste das erste Element, das den Index `0` hat, also die erste Textzeile. Schließlich wird die Liste der verbliebenen Zeilen (im Beispiel nur eine Zeile) mit dem `join`-Befehl zusammengefügt, wobei zwischen den Zeilen jeweils ein Umbruch `\n` eingefügt wird. Dies Verfahren funktioniert mit beliebig vielen Zeilen.

Folgendes Beispiel weist die Zeichenkette "Über Gott sagte Voltaire: Wenn es Gott nicht gäbe, müsste man Gott erfinden." einer Referenz namens `text1` zu und bestimmt, bei welchem Index in dieser Zeichenkette die Teilsequenz `"Gott"` zuerst auftritt. Die Indexzahl wird mit einer Referenz namens `pos` gespeichert. Dann wird in der Zeichenkette die Teilsequenz `"Gott"` durch `"jenes höhere Wesen, das wir verehren,"` ersetzt[5] und die neue Zeichenkette mit der Referenz `text2` gespeichert. Die Ergebnisse (die Werte von `pos` und `text2`) werden ausgegeben.

```
1 # Programm murke
2 text1 = "Über Gott sagte Voltaire: Wenn es Gott nicht\
3  gäbe, müsste man Gott erfinden."
4 pos=text1.find('Gott')
5 text2=text1.replace('Gott','jenes höhere Wesen, das\
6  wir verehren,')
7 print("pos = "+str(pos))
8 print(text2)
```

Formatierte Ausgabe von Zeichenketten

Bei der Ausgabe von Zeichenketten soll oft der Wert einer Variablen (der mit einer Referenz gespeichert ist) formatiert in die Zeichenkette eingebettet werden. Dies kann

[5] frei nach Heinrich Bölls „Doktor Murkes gesammeltes Schweigen“

mit der format-Methode geschehen, indem, anstelle der Variablen, geschweifte Klammern { } als Platzhalter eingefügt werden und der Name der Referenz der format-Methode als Argument übergeben wird. Wenn `p` den Wert `9.9` hat, ergibt zum Beispiel

```
print("Der Universalverrichter kostet nur {} Euro.".format(p))
```

die Ausgabe

```
Der Universalverrichter kostet nur 9.9 Euro.
```

In der geschweiften Klammer { } kann das Format in der Form `:b.nt` festgelegt werden, wobei `b` die minimale Ausgabebreite, `t` der Variablentyp (`f` Gleitkommazahl), `n` die Anzahl der (gerundeten) Nachkommastellen sind. So ergibt

```
print("Der Universalverrichter kostet nur {:6.2f} Euro.".format(p))
```

die Ausgabe

```
Der Universalverrichter kostet nur   9.90 Euro.
```

Die geschweiften Klammern, möglicherweise mit eingeschlossener Formatangabe, nennt man englisch *format field*. Die zu formatierende Zeichenkette kann beliebig viele format fields enthalten. Im Beispiel

```
print("{} und {} gingen in den Wald.".format("Hänsel", "Gretel"))
```

gibt es zwei format fields und zwei zugehörige Variablen. Es wird

```
Hänsel und Gretel gingen in den Wald.
```

ausgegeben. Die Variablen `Hänsel` und `Gretel` wurden der Reihe nach in die format fields eingesetzt. Es sind daher Positionsargumente (positional arguments). Der Einsatz der Positionsargumente kann auch durch natürliche Zahlen (beginnend mit 0) in den format fields bestimmt werden. Durch

```
print("{1} und {0} und nochmal {1}".format("Hänsel", "Gretel"))
```

erfolgt die Ausgabe von

```
Gretel und Hänsel und nochmal Gretel
```

Das erste Argument (`Hänsel`) hat den Index 0 und das zweite Argument (`Gretel`) hat den Index 1. Mehr zur Syntax dieser Zeichenkettenformatierung findet man auf der Webseite https://docs.python.org/3.10/library/string.html#formatstrings

Instanziierung eines Regulären Ausdrucks

Wenn man innerhalb einer langen Zeichenkette (String) nach einer genau bestimmten kürzeren Teilsequenz sucht, kann man diese als String darbieten. Will man dagegen nach einem Zeichenketten-Muster suchen, das auf ähnliche, aber verschiedene Teilsequenzen passt, nimmt man einen sogenannten Regulären Ausdruck (englisch: regular expression).

Um mit regulären Ausdrücken zu arbeiten, bindet man das Modul `re` ein, welches sie gleichnamige Klasse definiert.

```
import re
```

Ein Regulärer Ausdruck wird mithilfe einer Zeichenkette beschrieben. Der Backslash \ kann darin eine besondere Bedeutung haben (zur Bildung von Zeichenklassen, siehe unten) und soll nicht immer als Beginn einer Escape-Sequenz gedeutet werden. Daher wird die Zeichenkette für den Regulären Ausdruck als raw string geschrieben (siehe Seite 287). Allerdings behalten die üblichen Escape-Sequenzen, wie `\n`, ihre Bedeutung.

Die Funktion `re.compile` der Klasse `re`, welcher der raw string als Argument übergeben wird, erzeugt eine Instanz des Datentyps Regulärer Ausdruck. Zum Beispiel erzeugt

```
reo = re.compile(r"[aeiouäöü]")
```

eine Instanz des Typs Regulärer Ausdruck, kurz RE-Objekt, namens `reo`, welches das Zeichenkettenmuster *kleiner Vokal* darstellt, also einen beliebigen kleingeschriebenen Vokal darstellt.

Syntax Regulärer Ausdrücke

Ein Regulärer Ausdruck kann Zeichen enthalten, wie ein String, aber außerdem Zeichenklassen. Diese definieren eine Menge von Zeichen, von denen eines an dieser Stelle auftritt.

Eine Zeichenklasse kann durch eckige Klammern `[]` definiert werden. Wenn die Zeichenfolge in den eckigen Klammern nicht mit dem Zeichen ^ beginnt, können alle in den Klammern genannten Zeichen eingesetzt werden. Beispielsweise passt `C[DPS]U` auf `CDU`, `CPU` oder `CSU`.

Beginnt die Zeichenfolge in den eckigen Klammern mit dem Zeichen ^, dürfen die danach genannten Zeichen nicht eingesetzt werden. Beispielsweise passt `C[^DS]U` nicht auf `CDU` und `CSU`, aber auf alle anderen dreibuchstabigen Sequenzen, die mit C beginnen und mit U enden, zum Beispiel auf `CPU`.

Man kann in der Zeichenklassendefinition mithilfe des Zeichens – Bereiche von Buchstaben oder Zahlen wählen. Beispielsweise charakterisiert `[A-Z]ob` alle Wörter, die mit einem Großbuchstaben beginnen und mit `ob` enden, zum Beispiel `Aob` und `Bob`, nicht aber `bob` oder `8ob`.

Zeichenklassen kann man durch Sonderzeichen angeben, die diese Bedeutung nur in Regulären Ausdrücken haben. Einige davon werden durch zwei Zeichen, nämlich \ und ein Buchstabe, geschrieben. \n und \r behalten die Bedeutung der entsprechenden Escape-Sequenzen. Tabelle 8.12 führt einige Kurzbezeichnungen von Zeichenklassen auf.

Die Zeichenklasse \w enthält _ und alle alphanumerischen Zeichen, einschließlich Umlaute und ß, ist also gleich

```
[abcdefghijklmnopqrstuvwxyzäöüßABCDEFGHIJKLMNOPQRSTUVWXYZÄÖÜ01234567890_]
```

Beispiel zum Punkt: `x.z` passt auf `xyz` und `x§z`, nicht aber auf `xz`. Wenn der Reguläre Ausdruck einen Punkt . enthält, der nicht eine Zeichenklasse sondern ein gewöhnlicher Punkt sein soll, muss man `\.` schreiben. Entsprechendes gilt für andere Zeichen mit besonderer Bedeutung in Regulären Ausdrücken. Statt (und) muss man `\(` und `\)` schreiben.

`\d`	`[0-9]`	passt auf die Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8 und 9
`\s`	`[ \n\r\t\v\f]`	Whitespace-Zeichen (Leerzeichen und Zeilenvorschübe)
`\S`	`[^ \n\r\t\v\f]`	passt auf alle Zeichen, die nicht Whitespace-Zeichen sind
`\w`	`[a-zA-Z0-9]`	alphanumer. Zeichen und Unterstrich `_` , nicht & und +
`.`		Punkt . = Zeichenklasse, die auf jedes Zeichen passt
`\n`		Zeilenvorschub (wie in Python und Linux-Systemen üblich)

Tabelle 8.12: Zeichenklassen in Regulären Ausdrücken: Escape-Sequenzen und Punkt

Manchmal sollen Zeichen oder Zeichenklassen erkannt werden, die mehrfach hintereinander auftreten. Die Häufigkeit, mit der eine Zeichenklasse auftritt, kann mithilfe der in Tabelle 8.13 aufgeführten Quantoren angegeben werden.

`?`	kein- oder einmal	`[ab]?c` passt auf `c` , `ac` und `bc`
`*`	beliebig oft	`[ab]*c` passt auf `c` , `ac` , `bc` , `aac` , `abc` , `bac` , `bbc` ...
`+`	mindestens einmal	`[ab]+c` passt auf `ac` , `bc` , `aac` , `abc` , `bac` , `bbc` ... , nicht `c`
`{n}`	genau n mal	`ab{3}c` passt auf `abbbc` , nicht aber auf `abc abbc` , `abbbbc`
`{m, }`	mindestens m mal	`ab{2, }c` passt auf `abbc` , `abbbc` ... , nicht auf `ac` , `abc`
`{ ,n}`	höchstens n mal	`ab{ ,2}c` passt auf `ac` , `abc` , `abbc` , nicht `abbbc` , `abbbbc`
`{m,n}`	m bis n mal	`ab{1,2}c` passt auf `abc` und `abbc`, nicht `ac` , `abbbc`, `abbbbc`

Tabelle 8.13: Quantoren von Zeichenklassen in Regulären Ausdrücken

Zum Beispiel passt der Reguläre Ausdruck `[ab]{2}` , gleichbedeutend `[ab]{2,2}` , auf eine Sequenz aus genau zwei Zeichen der Menge {a,b}, also auf `aa` , `ab` , `ba` oder `bb` .

In einem Regulären Ausdruck kann eine zusammenhängende Menge von Zeichen oder Zeichenklassen mithilfe runder Klammern ( ) zu einer Gruppe zusammengefasst werden. Ein unmittelbar nach einer Gruppe stehender Quantor bezieht sich dann auf die ganze Gruppe. Zum Beispiel passt `(ab){1,2}c` nur auf `abc` und `ababc` , nicht aber auf `c` , `ac` , `bc` , `aac` oder `bbc` .

Eine oder-Verknüpfung wird mit dem Zeichen | (Pipe) geschrieben. Man kann sie mit Zeichen, Zeichenklassen oder Gruppen verwenden. Beispiel: `P(aul|eter)` passt auf `Paul` und `Peter` .

Methoden für Reguläre Ausdrücke

Die Methode `search` des RE-Objekts `r` sucht in einer Zeichenkette `A` das erste Auftreten bestimmter Teilsequenzen, beschrieben durch `r`. Ihr Argument ist `A`. Gibt es eine Übereinstimmung, wird ein Match-Objekt zurückgegeben, sonst `None` . Wurde ein Match-Objekt erzeugt, wird der eingerückte Befehl nach `if m:` ausgeführt, sonst nicht. Ein Match-Objekt `m` hat die Methoden `start` und `group` . `m.start()` gibt den Index des Beginns der Übereinstimmung als Ganzzahl (`int`) zurück und `m.group(0)`, oder etwas kürzer `m.group()`, gibt die übereinstimmende Teilsequenz als String (`str`) zurück.

`r.search(A)` `r = re.compile(r"[Jj]") ; r.search("Jena")` → *Match-Objekt*

Mit folgendem Python-Programm wird in einer Zeichenkette `A` nach dem ersten Auftreten eines Worts aus der Menge {equus, equum, asinus, asinum} gesucht.

```
# Programm dignitas
import re
A = "vere dignum et justum est, equum et salutare"
reo = re.compile(r"(equu[sm])|(asinu[sm])")
m = reo.search(A)
if m: print(m.group(),"bei Index " + str(m.start()))
else: print("kein Reittier gefunden")
```

Wird ein solches Wort gefunden, wird ein Match-Objekt `m` zurückgegeben, sonst `None`. Wurde ein Match-Objekt zurückgegeben, wird der Befehl nach `if m:` ausgeführt, sonst nicht. Dieser Befehl gibt das Wort aus und außerdem den Index der Zeichenkette `A`, bei dem das Wort beginnt.

Die Methode `findall` liefert eine Liste aller auf `reo` passenden Teilsequenzen, wenn die untersuchte Zeichenkette `A` keine Gruppe enthält.

`r.findall(A)` `r=re.compile(r"\d") ; r.findall("1or2")` → `['1', '2']`

Sind jedoch Gruppen enthalten, werden die auf die Gruppen passenden Teilsequenzen in einer Liste zurückgegeben.

Die Methode `finditer` gibt einen Iterator für die Match-Objekte zurück. In einer for-Schleife kann mit der Methode `group()` auf die übereinstimmenden Teilsequenzen zugegriffen werden.

`r.finditer(A)` `r=re.compile(r"(\S)\1") ; r.finditer("aaii")` → Iterator

Die Methode `sub` ersetzt in der Zeichenkette `A` alle auf `r` passenden Teilsequenzen durch eine Zeichenkette `s`, mit `s` und `A` als Argumente. Rückgabewert ist die veränderte Zeichenkette.

`r.sub(s,A)` `r=re.compile(r"[aeiouäöü]") ; r.sub("i","Jena")` → `"Jini"`

Kurznachrichten von der Webseite https://www.tagesschau.de/ werden mit folgendem Python-Programm gezeigt.

```
# Program ts.py to show Tagesschau news
import requests, re, subprocess, textwrap
url = "https://www.tagesschau.de/"
r = requests.get(url)
re1 = re.compile(r"teaser__shorttext\">\s+(.*?)\s+</p>")
re2 = re.compile(r"<.*?>(.*)</.*?>")
```

```
7  lst = re1.findall(r.text)
8  txt = "Nachrichten der Tagesschau (Taste q zum Beenden)"
9  for i in lst:
10     par = textwrap.fill(re2.sub(r"\g<1>",i), width=60)
11     txt = txt + "\n\n" + par
12 with open("ts.txt", "w") as f:
13     f.write(txt)
14 subprocess.run(["less", "ts.txt"])
```

Die Module `requests`, `subprocess` und `textwrap` werden später vorgestellt (siehe Seiten 335, 341 beziehungsweise 293). In Zeilen 3 – 4 wird die Webseite geöffnet. Die Referenz `r.text` verweist auf den HTML-Quellcode. In Zeile 5 wird ein regulärer Ausdruck definiert, der auf jede einzelne Kurznachricht im HTML-Quellcode passt. In Zeile 6 wird ein regulärer Ausdruck definiert, der auf Zeichenketten passt, die ein öffnendes und ein schließendes HTML-Tag enthalten. Die Zeichenkette zwischen den Tags bildet Gruppe 1. In Zeile 7 wird eine Liste `lst` gebildet, die alle Kurznachrichten enthält. In Zeile 8 wird eine Titelzeile gebildet und der Referenz `txt` zugewiesen. In Zeilen 9 – 11 wird der Nachrichtentext in Absätze mit Zeilen von höchstens 60 Zeichen zerlegt, damit er besser lesbar ist, und an die Titelzeile `txt` angehängt. Dabei werden in Zeile 10 öffnende und ein schließende HTML-Tags entfernt. In Zeilen 12 – 13 wird der formatierte Text in die Datei `s.txt` geschrieben. In Zeile 14 wird der terminal pager `less` aufgerufen, um die Datei `s.txt` auf dem Bildschirm anzuzeigen. Drücken der Taste `q` beendet `less`.

8.2 Python-Module

8.2.1 Text, Datum und Zeit, Kalender

String-Konstanten im Modul string

Das Modul `string` enthält einige Konstanten vom Typ `str` (Zeichenketten), darunter `string.digits` mit den Ziffern, also `"0123456789"`, und `string.whitespace` mit allen whitespace-Zeichen. Damit kann man alle whitespaces aus einer Zeichenkette entfernen:

```
1 import string
2 txt = " MALWMRLLPL  LALLA\n LWGPD   PAAAF VNQHL"
3 for wsp in string.whitespace:
4     txt = txt.replace(wsp,"")
5 print(txt)
```

ergibt `MALWMRLLPLLALLALWGPDPAAAFVNQHL`

Textblöcke formatieren mit textwrap

Das Modul `textwrap` vereinfacht die Formatierung von Textblöcken, unter anderem mit den Methoden `fill` und `wrap`.

Die Methode `fill` packt einen Text, der als string (Zeichenkette) gegeben ist, in einen Absatz mit bestimmter Spaltenbreite (= maximale Zeilenlänge), indem Zeilenumbrüche `\n` eingefügt werden, und gibt ihn als string (Zeichenkette) zurück.

```
1 import textwrap
2 txt = "the price for independence"
3 print( textwrap.fill(txt,width=10) )
```

ergibt

```
the price
for indepe
ndence
```

Die Methode `wrap` packt, ebenso wie `fill`, einen Text in einen Absatz mit bestimmter Spaltenbreite, fügt aber keine Zeilenumbrüche ein und gibt eine Liste (Datentyp `list`) zurück, deren Elemente die Zeichenketten der Zeilen sind.

```
1 import textwrap
2 txt = "the price for independence"
3 print( textwrap.wrap(txt,width=10) )
```

ergibt

```
['the price', 'for indepe', 'ndence']
```

Im folgenden Beispiel wird eine Zeichenkette aus Buchstaben, welche eine Aminosäuresequenz darstellt (`raw`), mithilfe der Methode `wrap` in Elemente zerlegt, die höchstens 10 Zeichen enthalten (Zehnergruppen). Die resultierende Liste wird mit der Methode `join` zu einer Zeichenkette (`div`) zusammengefügt, wobei benachbarte Zehnergruppen durch ein Leerzeichen `" "` getrennt werden. Anschließend wird `div` mit der Methode `fill` in Zeilen mit einer Breite von höchstens 32 Zeichen zerlegt (`box`) und ausgegeben.

```
1 import string, textwrap
2 raw = "MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGE"
3 div = " ".join( textwrap.wrap(raw,width=10) )
4 box = textwrap.fill(div,width=32) ; print(box)
```

ergibt

```
MALWMRLLPL LALLALWGPD PAAAFVNQHL
CGSHLVEALY LVCGE
```

Modul docx für Dokumente im Format .docx

Bürokraten arbeiten oft mit dem Textverarbeitungsprogramm WORD der Firma Microsoft und dem Dateiformat Office Open XML Document, erkennbar an der Dateiendung `.docx`. Wie man aus einer Datei `word.docx` eine PDF-Datei macht, die man mit dem Programm `fbgs` auf dem Bildschirm zeigen kann, wurde im Abschnitt *Wandlung von .docx* auf Seite 236 beschrieben.

Das Modul `docx` kann WORD-Dateien im Dateiformat Office Open XML Document lesen und schreiben. Es wird mit

```
pip3 install --user python-docx
```

installiert und mit `import docx` in einem Pythonprogramm eingebunden.

Folgendes Programm liest eine WORD-Datei (`word.docx`) und speichert den Textinhalt ohne Auszeichnungen (wie Fettdruck oder Unterstreichung) und ohne eventuell enthaltene Bilder als einfache Textdatei (`word.txt`).

```
1  # Program wordread.py word.docx -> word.txt
2  import docx  # was installed as python-docx
3  import textwrap  # module for breaking long lines
4  ein = "word.docx" ; aus="word.txt"
5  doc = docx.Document(ein)
6  print(ein, "with", len(doc.paragraphs), "paragraphs")
7  lcp = []  # list for the content of the paragraphs
8  for p in doc.paragraphs:
9      t = p.text.strip()  # par.text without whitespaces
10     if len(t) > 0:      # exclude empty lines
11         t = textwrap.fill(t,width=80)  # max. 80 char.
12         lcp.append(t)   # append the formatted content
13 txt = "\n\n".join(lcp)  # text content of file
14 with open(aus,"w") as file: file.write(txt)
15 print("written as a simple text file", aus)
```

Alle Absätze (paragraphs) werden mithilfe des Moduls `textwrap` auf eine maximale Zeilenlänge von 80 Zeichen umgebrochen. Zwischen Absätzen wird eine Leerzeile eingefügt.

Folgendes Programm schreibt eine WORD-Datei (`word.docx`), deren Dokument aus einer Titelzeile, einer Untertitelzeile (Autor und Datum) und zwei Absätzen besteht. Zwischen den Absätzen ist ein Bild (`Titanic.png`) eingefügt.

```
1  # Program wordwrite.py word.txt -> word.docx
2  import docx  # was installed as python-docx
3  from docx.oxml.ns import qn
4  from docx.enum.text import WD_ALIGN_PARAGRAPH
5  #
6  h1="Triumph der Technik"
7  h2="Max von Überheblich (2. April 1912)"
8  p1="Ein moderner Passagierdampfer bietet den \
9  gleichen Komfort wie ein gutes Hotel auf dem \
10 dem Festland und bewegt sich dabei übers Meer."
11 i1='/home/ahg/image/Titanic.png' # image file
12 p2="Die Technik des zwanzigsten Jahrhunderts \
13 erbringt den glänzenden Sieg der Menschheit \
14 über geistlose Gewalten der unbelebten Natur."
15 #
```

```
16 doc = docx.Document()  # empty docx object
17 # page size and margins (Seitenränder) in mm
18 sec = doc.sections[0]  # section 0 of document
19 sec.page_height = docx.shared.Mm(297)  # A4
20 sec.page_width  = docx.shared.Mm(210)  # A4
21 sec.left_margin   = docx.shared.Cm(2.5)
22 sec.right_margin  = docx.shared.Cm(2.5)
23 sec.top_margin    = docx.shared.Cm(2.0)
24 sec.bottom_margin = docx.shared.Cm(2.0)
25 # font, font size, line spacing (Zeilenabstand)
26 fnt = "Arial" # font (Zeichensatz) for document
27 sty = doc.styles["Normal"]
28 sty.font.name = fnt  # font (Zeichensatz)
29 sty.font.size=docx.shared.Pt(12)  # (Kegelhöhe)
30 sty.paragraph_format.line_spacing = 1.3
31 #
32 tit = doc.add_heading(h1, 0)  # title (Titel)
33 rFonts = tit.style.element.rPr.rFonts
34 rFonts.set(qn("w:asciiTheme"), fnt)
35 sub = doc.add_heading(h2, 3)  # header 3
36 rFonts = sub.style.element.rPr.rFonts
37 rFonts.set(qn("w:asciiTheme"), fnt)
38 pre = doc.paragraphs[-1] # previous paragraph
39 pre.alignment = WD_ALIGN_PARAGRAPH.CENTER
40 doc.add_paragraph()  # empty space (Leerraum)
41 doc.add_paragraph(p1)  # paragraph (Absatz)
42 doc.add_picture(i1, width=docx.shared.Cm(10))
43 pre = doc.paragraphs[-1]  # previous paragraph
44 pre.alignment = WD_ALIGN_PARAGRAPH.CENTER
45 doc.add_paragraph(p2)  # paragraph (Absatz)
46 doc.save("word.docx")  # docx object -> file
```

Das Papierformat ist A4. Die Seitenränder sind links und rechts jeweils 2,5 cm, oben und unten 2,0 cm. Als Schriftart wurde einheitlich `Arial` gewählt. In den Absätzen beträgt die Schriftgröße (Kegelhöhe) 12 pt und der Zeilenabstand 1,3 = 130 %. Das eingefügte Bild wurde bei gleichbleibendem Seitenverhältnis auf eine Breite von 10 cm skaliert.

Archivierung: tarfile

Das Modul `tarfile` dient der Archivierung einer Datei oder eines Verzeichnisses in komprimierter Form. In folgendem Beispiel wird das Verzeichnis `/home/ahg/texts/` zunächst in eine Archivdatei ohne Kompression (tarball) gepackt und diese dann mittels *gzip* (GNU zip) komprimiert und in der neuen Datei `texts_`*datumzeit*`.tar.gz` (mit der Referenz `raus`) im Arbeitsverzeichnis gespeichert. Dabei ist *datumzeit* eine Zeichenkette

der Länge 12, die Datum und Uhrzeit der Speicherung darstellt, siehe Seite 299.

```
1 import os.path, tarfile, time
2 rein = "/home/ahg/texts/"
3 zeit = time.strftime("%y%m%d%H%M%S")
4 raus = "texts_"+zeit+".tar.gz"
5 arch = os.path.basename(rein)
6 with tarfile.open(raus, "w:gz") as t:
7     t.add(rein, arcname=arch)
```

Die Funktion `os.path.basename` in Zeile 5 gibt das Ende eines Pfades zurück (englisch base, das ist der Teil nach dem letzten `/`, der andere Teil heißt Kopf). Für ein Verzeichnis, wie in obigem Beispiel, ist das ein leerer string `""` und für eine Datei ist das der Dateiname ohne die übergeordneten Verzeichnisse. In Zeile 6 wird die komprimierte Archivdatei, deren Name durch `raus` referenziert wird, zum Schreiben geöffnet. Zeile 7 fügt die durch `rein` angegebene Instanz rekursiv dem Archiv hinzu. Rekursiv bedeutet, dass bei einem Verzeichnis auch der Inhalt der Unterverzeichnisse berücksichtigt wird. Bei einem Verzeichnis ist `arch = ""` und die Option `arcname=arch` bewirkt, dass die Unterverzeichnisse und Dateien ohne den Kopf des Pfades (in obigem Beispiel: ohne `/home/ahg/texts/`) rekursiv archiviert werden. Dagegen ist bei der Archivierung einer einzelnen Datei `arch` gleich dem Dateinamen ohne Kopf des Pfades und die Datei wird so archiviert.

Im folgenden Beispiel wird der Inhalt einer Archivdatei ausgepackt.

```
1 import tarfile
2 teer = "texts_220206181109.tar.gz"
3 with tarfile.open(teer, "r:gz") as t:
4     t.extractall("texts/")
```

Der Archivinhalt wird im Verzeichnis `texts/` im Arbeitsverzeichnis gespeichert.

Datum und Zeit: time, Dateinamen mit Zeitabgabe

Zeitangaben werden in einem time standard definiert, der Nullpunkt und Längeneinheit der Zeitskala festlegt, sowie Verschiebungen aus besonderen Gründen (zum Beispiel durch Schaltsekunden, englisch: leap seconds). Der wichtigste time standard heißt *koordinierte Weltzeit* (cordinated universal time), UTC. Er wird auch in Linux verwendet. Die koordinierte Weltzeit UTC gibt die Zeit für jeden Ort der Erde einheitlich an.[6]

Zusätzlich zur UTC wird für jeden Ort eine lokale Zeit gebildet, die sich nach bestimmten Regeln aus der UTC berechnen lässt. In Deutschland heißt die lokale Zeit im Winter mitteleuropäische Zeit MEZ oder mitteleuropäische Normalzeit (englisch: central european time, CET) und im Sommer mitteleuropäische Sommerzeit MESZ (englisch:

[6] Wir sehen hier von Spitzfindigkeiten der Relativitästheorie ab.

central european summer time, CEST).[7] Bei Zeitangaben sollte man erklären, ob sie UTC oder die lokale Zeit darstellen. Wir geben meistens die lokale Zeit an.

Intern wird die Zeit vom Betriebssystem aber in Form der *Unix time* dargestellt. Die aktuelle Unix time ist in der Regel gleich der Differenz zwischen der aktuellen UTC und der UTC des willkürlich gewählten Zeitpunkts 1. Januar 1970, 0 Uhr, in Sekunden.[8] In der Unix time wird die Einheit (Sekunde) weggelassen. Sie ist also eine Zahl ohne Einheit. Die Unix time ist für Zeiten vor 1970 negativ. Ein Zahlenwert, der eine Unix time darstellt, heißt *Unix timestamp*.

Das Modul `time` liefert die aktuelle Zeit und erlaubt deren Formatierung als string (Zeichenkette). Außerdem ermöglicht es die Unterbrechung der Programmausführung für eine bestimmte Dauer.

Wird die Methode `localtime` ohne Argumente aufgerufen, gibt sie eine Instanz vom Typ `time.struct_time` zurück, welche die aktuelle lokale Zeit beschreibt. Entsprechend liefert die Methode `gmtime` eine Instanz vom Typ `time.struct_time` für UTC.

Objekte vom Typ `time.struct_time` haben Attribute, mit denen auf Werte in verschiedenen Zeiteinheiten zugegriffen werden kann. Bei einigen kann man alternativ über einen Index zugreifen, wie bei sequentiellen Datentypen., siehe Tabelle 8.14.

Attribut	Index	Bedeutung
tm_year	0	Jahr (zum Beispiel 1959)
tm_mon	1	Monat (1 – 12)
tm_mday	2	Tag des Monats (1 – 31)
tm_hour	3	Stunde (0 – 23)
tm_min	4	Minute (0 – 59)
tm_sec	5	Sekunde (0 – 61)
tm_wday	6	Wochentag (0 – 61), 0 ist Montag
tm_yday	7	Tag im Jahr (0 – 366)
tm_isdst	8	0 Winterzeit, 1 Sommerzeit, -1 unbekannt
tm_zone		Abkürzung für die Zeitzone (zum Beispiel CET)
tm_gmtoff		Abweichung von UTC in Sekunden

Tabelle 8.14: Attribute von Instanzen des Datentyps `time.struct_time`. Alle, bis auf `tm_zone` (str), haben den Datentyp `int`.

Folgendes Programm gibt die Zeitzone (in Deutschland CET oder CEST) aus und zudem, ob standard time (Normalzeit) oder daylight savings time (Sommerzeit) gilt.

[7] Der Grund für die Einführung der lokalen Zeit MEZ war, vereinfachend gesagt, dass das Mittagessen um 12 Uhr auf dem Tisch stehen sollte. Die Beibehaltung der lokalen Zeit ist, vereinfachend gesagt, ein bewährtes Mittel zur Verwirrung der Untertanen.

[8] Der UTC Zeitpunkt 1. Januar 1970, 0 Uhr heißt *Unix epoch*. Für Rosinenausscheider sei angemerkt, dass die aktuelle Unix time während der Schaltsekunden der UTC von der Differenz zwischen der aktuellen UTC und der UTC der Unix epoch abweicht. Aber das ist in der Praxis unbedeutend, denn das Betriebssystem kann immer zwischen UTC und Unix time korrekt umrechnen.

```
import time
z = time.localtime().tm_zone
d = time.localtime().tm_isdst
print("Time zone:", z)
if d==0: print("standard time")
if d==1: print("daylight savings time")
```

Die Funktion `time.strftime` wandelt eine Instanz vom Typ `time.struct_time`, welche als zweites Argument übergeben werden kann, in einen string, gemäß einer Formatangabe, die als erstes Argument übergeben wird. Ohne zweites Argument wird die `time.struct_time` Instanz automatisch mit der Methode `localtime` für die aktuelle lokale Zeit gebildet.

Folgendes Programm gibt Datum und Uhrzeit aus. Deutsche Schreibweise wird erreicht, indem die lokale Spracheinstellung des Betriebssystems übernommen wird.

```
# Programm zur Ausgabe von Datum und Zeit
import locale, time
locale.setlocale(locale.LC_ALL,"")
f1 = "%A, %d. %B %Y"
datum = time.strftime(f1)
f2 = "%H Uhr %M und %S Sekunden"
uhrzeit = time.strftime(f2)
print(datum)
print(uhrzeit)
```

Eine Zeichenkette, die Datum und Zeit ohne Sonderzeichen und ohne Leerzeichen als zwölfstellige Zahl darstellt, wird folgendermaßen erzeugt.

```
#!/usr/bin/python3
import time
zeitform = "%y%m%d%H%M%S"
zeittext = time.strftime(zeitform)
print(zeittext)
```

Damit kann man zum Beispiel einen Dateinamen bilden, der die Zeit enthält, zu der die Datei gespeichert wurde. Der Dateiname wird so eindeutig.

Man kann einen Zeitpunkt auch durch die Anzahl der Nanosekunden angeben, die seit der Unix epoch (01.01.1970, 0 Uhr) vergangen sind.

```
import time
t_ns = time.time_ns()
print(t_ns)
```

Das Ergebnis ist eine ganze Zahl mit 19 Stellen. Der Datentyp von `t_ns` ist `int`.

Eine Zeitmessung benötigt man, um die Dauer einer langwierigen Berechnung innerhalb eines Programms zu messen. Folgendes Beispiel enthält, statt der Rechnung, eine Pause von 3 s, während der das Programm angehalten ist. Wir bestimmen die Zeit vor und nach der Pause und geben die Differenz aus.

```
1 #!/usr/bin/python3
2 import time
3 z1 = time.time() # Unixzeit vor der Pause
4 time.sleep(3) # Programm pausiert für 3 s
5 z2 = time.time() # Unixzeit nach der Pause
6 print("Pausendauer: " + str(z2-z1) + "s")
```

Die Methode `sleep` des `time` Moduls wird häufig gebraucht. Beispiele sind ein Programmm das in einer unendlichen Schleife auf Nutzereingaben wartet, oder ein Programm, das eine Leuchtdiode blinken lässt (für Abstand und Dauer der Leuchtperioden).

Kalender und Datumsinformation: calendar

Mit dem Modul calendar kann man einen Kalender ausgeben oder Information zum Datum erhalten. Mit folgendem Programm erfolgt die Ausgabe im Terminalfenster:

```
1  #!/usr/bin/python3
2  #  Ausgabe vom Kalendermonat und Wochentag des 24. 12.
3  import calendar, time
4  #
5  dtu="de_DE.utf8" # Locale (Sprache_Region.Kodierung)
6  jr=int(time.strftime("%Y")) # aktuelle Jahreszahl
7  mo=int(time.strftime("%m")) # aktuelle Monatszahl
8  calendar.LocaleTextCalendar(locale=dtu).prmonth(jr,mo)
9  #
10 wtd=["Montag","Dienstag","Mittwoch","Donnerstag",\
11 "Freitag","Sonnabend","Sonntag"]
12 wot=calendar.weekday(jr,12,24) # 0 <= wot <= 6
13 print("Heiligabend fällt auf einen "+wtd[wot]+".")
```

Das aktuelle Jahr und der Monat werden über das Modul time geholt und der aktuelle Monat mit einer Methode des Moduls calendar ausgegeben. Eine andere Methode berechnet den Wochentag, auf den Heiligabend in diesem Jahr fällt.

Folgendes Beispiel zeigt einen Kalender für den Monat April 2029.[9]

[9] Am Freitag, 13. April 2029, nähert sich Asteroid Apophis unserer Erde und verfehlt sie nur knapp. – Hoffentlich. – Experten raten zur Vorsicht: Meiden Sie den Planeten Erde an diesem Tag!

```
#!/usr/bin/python3
import calendar
c = calendar.LocaleTextCalendar()
print(c.formatmonth(2029,4)) # Jahr, Monat
```

Ohne Argumente verwendet die Methode `LocaleTextCalendar()` für die Ausgabe die im Betriebssystem festgelegte Spracheinstellung (locale) des Nutzers (hier: Deutsch).

8.2.2 Daten, Zahlen und Statistik, Graphen

csv

Beim Austausch von strukturierten Daten zwischen verschiedenen Computerprogrammen, auch über das Internet, muss das Format, einschließlich der Anordnung der Daten, festgelegt werden. Computerprogramme brauchen Zugriff auf ausgesuchte Teile der Daten und es es ist vorteilhaft, wenn sie für Menschen leicht lesbar sind. Häufig verwendete Datenformate sind CSV und JSON (siehe unten).

CSV dient der Speicherung von Tabellen und ähnlichen Datenstrukturen. Eine CSV-Datei sollte die Endung `.csv` haben. CSV bildet die Zeilen einer Tabelle als Zeilen der Datei ab. Die Tabellenfelder werden zu Zeichenketten. Jede Zeile der CSV-Datei enthält daher in der Regel so viele dieser Zeichenketten, wie es Spalten in der Tabelle gibt.

Die Zeichenketten der Tabellenfelder werden in der CSV-Datei durch ein besonderes Trennzeichen getrennt. Meistens wird das Komma verwendet, daher der Name CSV. Dann dürfen in den Zeichenketten der Tabellenfelder zwei Zeichen nicht vorkommen: Komma und Zeilenendezeichen (newline).

Ein Problem entsteht, wenn das Komma in Tabellenfeldern vorkommt, zum Beispiel als Teil einer Zeichenkette. Mögliche Lösungen sind: a) Wahl eines anderen Trennzeichens (zum Beispiel Semikolon oder Pipe-Teichen `|`), b) Einfassung der Zeichenkette, die das Komma enthält, in ein Paar von englischen Anführungszeichen (`"`).

Aus oben genannten Gründen gibt es verschiedene Varianten von CSV. Zum Lesen und Schreiben von CSV-Dateien dient das Python-Modul `csv`. Es erlaubt die Wahl einer CSV-Variante (dialect). Wichtige Varianten sind `excel` (Trennzeichen = Komma) und `excel-tab` (Trennzeichen = Tabulator, sonst wie `excel`) für die Freunde unfreier Software und `unix` für Nutzer von Linux und anderen UNIXartigen Betriebssystemen. Ohne Angabe der CSV-Variante ist standardmäßig `dialect="excel"`.

Schreiben einer CSV-Datei erfordert zunächst das entsprechende Öffnen einer Textdatei (siehe Seite 272). Die Funktion `writer` des Moduls `csv` erzeugt ein writer-Objekt, welches die Methoden `writerow` und `writerows` hat. Sie schreiben eine einzelne Zeile beziehungsweise mehrere Zeilen auf einmal. Das Argument von `writerow` ist eine Liste (oder ein anderes iterierbares Objekt). Die Liste muss Zeichenketten oder Referenzen auf Zeichenketten enthalten. Das Argument von `writerows` ist eine Liste von Listen (oder allgemeiner ein iterierbares Objekt, das aus iterierbaren Objekten besteht).

In folgendem Beispiel wird eine CSV-Datei `datei.csv` in der CSV-Variante (dialect)

unix geschrieben. Zunächst wird mit writerow die Zeile der Spaltenköpfe geschrieben, dann mit writerows die Datensätze.

```
# Programm "csv_write" mit Python 3, AHG (2020)
import csv
with open("datei.csv","w") as f:
    wob = csv.writer(f,dialect="unix")
    wob.writerow(["Jahr","Vorname","Nachname"])
    wob.writerows([
    ["1949","Konrad","Adenauer"]
    ["1963","Ludwig","Erhard"]
    ])
```

Das Ergebnis ist eine Datei datei.csv, in der ein Komma das Trennzeichen ist und alle Feldelemente als Zeichenketten in doppelten Anführungszeichen stehen (auch wenn bei der Eingabe einfache Anführungszeichen verwendet wurden oder Zahlen ohne Anführungszeichen standen, zum Beispiel 1949 statt "1949".)

Lesen einer CSV-Datei erfordert ein Dateiobjekt, das für Lesezugriff geöffnet wurde. Die Funktion reader des Moduls csv erzeugt ein reader-Objekt. Dieses implementiert das Iteratorprotokoll und kann darum mit einer for-Schleife durchlaufen werden (siehe Seite 277). In der for-Schleife liefert das reader-Objekt für jede Zeile der CSV-Datei eine Liste von einzelnen Zeichenketten (Datentyp str), die jeweils einem Tabellenfeld entsprechen.

In folgendem Beispiel wird die CSV-Datei datei.csv, die in obigem Beispiel in der CSV-Variante (dialect) unix geschrieben wurde, gelesen und der Inhalt zeilenweise (jeweils als Liste) ausgegeben.

```
# Programm "csv_read" mit Python 3, AHG (2020)
import csv
with open("datei.csv","r") as f:
    rob = csv.reader(f,dialect="unix")
    for zeile in rob:
        print(zeile)
```

Das reader-Objekt mit dialect="unix" erkennt die Zeichenketten in der CSV-Datei (Tabellenfelder), wenn sie in doppelten Anführungszeichen stehen und auch wenn sie ohne Anführungszeichen stehen. Allerdings dürfen in der CSV-Datei keine Leerzeichen eingefügt werden, die nicht Teil der Zeichenkette sind.

Da ein normierter Standard für das Formats von CSV-Dateien fehlt, ist zwar das Lesen einer CSV-Datei, die man selbst geschrieben hat, problemlos. Das Lesen einer CSV-Datei, die aus anderer Quelle stammt und deren CSV-Variante (dialect) man nicht kennt, kann jedoch schwierig sein. Man müsste zum Beispiel wissen, welches Trennzeichen verwendet wird. Hier kann die Methode sniff() helfen, die von der Sniffer-Klasse

des Moduls `csv` bereitgestellt wird. Sie ermittelt die CSV-Variante (einschließlich Trennzeichen) automatisch anhand einer Teilmenge der CSV-Datei (Beispiel: die ersten 1024 Bytes). Die Anwendung wird in folgendem Beispiel gezeigt.

```
1 # Programm "csv_sniff" mit Python 3, AHG (2020)
2 import csv
3 with open("datei.csv","r") as f:
4     dlt = csv.Sniffer().sniff(f.read(1024))
5     f.seek(0) # Zeiger auf Anfang der Datei
6     rob = csv.reader(f,dialect=dlt)
7     for zeile in rob:
8         print(zeile)
```

Die Methode `seek(0)` wurde in Tabelle 8.6 auf Seite 272 vorgestellt.

json

Neben CSV ist JSON (JavaScript Object Notation) ein Dateiformat von Textdateien zur Speicherung strukturierter Daten. Es besteht aus einer Zeichenkette, die dem Dictionary von Python ähnelt. Die Umwandlung JSON ↔ Dictionary ist daher einfach. Ebenso wie Python benutzt JSON normalerweise UTF-8 zur Zeichencodierung und das Komma als Aufzählungsoperator.

JSON kennt folgende Datentypen: Dezimalzahl (number), Zeichenkette (string, eingeschlossen in doppelte Anführungszeichen, Escape-Sequenzen mit \ ), boolesche Werte (`true` und `false`, klein geschrieben), Feld (array, eine geordnete Liste beliebiger Objekte in [ ]) und Objekte (objects) mit Schlüssel-Wertpaaren in { }. Der Schlüssel eines Objekts ist eine Zeichenkette. Nach : folgt der zugehörige Wert (beliebiger Datentyp). Außerdem gibt es den Datentyp `null` (entspricht `None` in Python). Tabelle 8.15 zeigt die Übersetzung von Datentypen oder `Instanzen` aus Python in JSON-Elemente.

Python	JSON
dict	object
`list`	array
`tuple`	array
`str`	string
int	number
float	number
`True`	`true`
`False`	`false`
`None`	`null`

Tabelle 8.15: Python ↔ JSON

Ein JSON-Objekt ist geordnet; die Reihenfolge der Elemente ist nicht zufällig. Ebenso ist in Python seit Version 3.7 gewährleistet, dass die Reihenfolge der Schlüssel in einem `dictionary` erhalten bleibt.

Eine häufige Aufgabe in der Informatik ist die automatische Zerlegung einer Zeichenkette in Begriffsteile mit eigenständiger Bedeutung. Diesen Vorgang nennt man Parsing, das Verb ist parsen und ein Computerprogramm, welches parst, ist ein Parser. Das Wort stammt vom lateinischen `pars` (Femininum), das Teil. Ein Vorteil von einem JSON-Objekt ist, ebenso wie bei einem Python `dictionary`, dass es leicht zu parsen ist. JSON-Objekte können überdies von vielen verschiedenen Programmiersprachen verarbeitet werden.

Um in einem Pythonprogramm mit JSON zu arbeiten, lädt man das Modul `json`.

Das folgende Programm erzeugt ein dictionary `daten_dict`, in welchem dem key `"MT.1.257"` als value ein dictionary zugeordnet wird. Das dictionary `daten_dict` wird mit der Funktion `dump` als Zeichenkette im JSON Format in einer Datei gespeichert. JSON-Dateien haben üblicherweise die Endung `.json`.

```
1 #!/usr/bin/python3
2 import json
3 daten_dict = {
4     "MT.1.257": {
5         "Modulname": "Bioinformatik",
6         "Koordinator": "Prof. Gitter"
7              }  }
8 with open("daten.json", "w") as datei:
9     json.dump(daten_dict, datei)
```

Die Datei namens `daten.json` wurde mit (`w`) zum Schreiben im `text mode` geöffnet und einem file object (handle) namens `datei` zugeordnet. Die Funktion `dump` hat zwei Argumente: das dictionary, welches serialisiert werden soll, und das file object, in welches geschrieben wird. Serialisierung ist die Umwandlung einer strukturierten Instanz in einen sequenziellen Datentyp. Obwohl die Zeichenkette ein JSON.Format enthält, hat sie in Python den sequenziellen Datentyp `str`.

Hat man eine Datei mit JSON, kann man den Inhalt (eine Zeichenkette), nach Öffnen der Datei, mit Funktion `load` lesen und in ein dictionary umwandeln (deserialisieren). Das folgende Programm öffnet die im vorigen Programm erzeugte Datei, deserialisiert und gibt ein dictionary zurück. Mit der Funktion `dumps` (mit `s`) wird dann aus dem dictionary wieder eine Zeichenkette (Datentyp `str`), welche aber nicht wieder in einer Datei gespeichert, sondern mit der Funktion `print` auf dem Bildschirm ausgegeben wird.

```
1 #!/usr/bin/python3
2 import json
3 with open("daten.json", "r") as datei:
4     daten_dict = json.load(datei)
5 print(json.dumps(daten_dict, indent=4))
```

Die Datei namens `daten.json` wurde mit (`r`) zum Lesen im `text mode` geöffnet und einem file object namens `datei` zugeordnet. Während die Funktion `dump` dazu dient, in eine Textdatei zu schreiben, wird mit der Funktion `dumps` eine Zeichenkette zur weiteren Verwendung im Pythonprogramm erzeugt. Diese kann, wie im Beispiel, am Bildschirm ausgegeben werden, oder mit anderen Funktionen verarbeitet werden. Die Option `indent=4` bewirkt die Einfügung von Zeilenumbrüchen und Einrückungen (Vielfache von 4 Leerzeichen), um die Ausgabe am Bildschirm für Menschen besser lesbar zu machen.

Mit der Funktion `loads` (mit `s`) wird eine Zeichenkette im JSON-Format gelesen und (anders als bei `load`) in einer Zeichenkette (statt in ein dictionary) gespeichert.

Molekülmassen: periodictable

Das Modul `periodictable` mit Daten der chemischen Elemente wird mit dem Befehl

```
sudo apt install python3-periodictable
```

installiert. Man kann unter anderem die molekulare Masse (englisch: molecular mass) einer chemischen Verbindung berechnen, wie folgendes Beispiel für H_2O zeigt.

```
1 import periodictable as pt
2 print("atomic mass of elements")
3 for e in [ pt.H, pt.O ]:
4     print(e.name, e.symbol, e.mass, "Da")
5 print("molecular mass of compound")
6 c = pt.formula("H2O")
7 print(c, c.mass, "Da")
```

Es gibt eine Webseite mit der Dokumentation des Moduls.

random

Das Modul random stellt Zufallszahlen bereit. Eine Liste ganzzahliger Zufallszahlen aus einem bestimmten Zahlenbereich, ohne zweimaliges Auftreten derselben Zahl, liefert folgende Programm:

```
1 #!/usr/bin/python3
2 # Liste ganzzahliger Zufallszahlen ohne Wiederholung
3 import random
4 #
5 n = 6 # Anzahl der Werte, die erzeugt werden
6 a, e = 1, 49 # kleinst- und größtmögliche Zahl
7 zow = random.sample(range(a,(e+1)),n) # Zahlen-Liste
8 # Liste von random.sample ohne identische Elemente
9 print(zow)
```

Jeder Aufruf des Programms erzeugt eine neue Liste mit Zufallszahlen.

statistics

Das Modul statistics enthält Funktionen für einfache beschreibende Statistik reeller Werte. Unter anderem

- `mean()` arithmetisches Mittel (Mittelwert)
- `geometric_mean()` geometrisches Mittel
- `median()` Median (Zentralwert)
- `pvariance()` Varianz einer Grundgesamtheit
- `pstdev()` Standardabweichung einer Grundgesamtheit
- `variance()` Varianz (aus einer Stichprobe)
- `stdev()` Standardabweichung (aus einer Stichprobe)

Folgendes Programm berechnet statistische Werte anhand einer Stichprobe:

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
# Python berechnet
import math, statistics
dat = [11.25, 4.75, 7.0, 5.25, 12.5, 7.75] # Stichprobe
mwt = str( round( statistics.mean(dat), 1) )
med = str( round( statistics.median(dat), 1) )
ssd = str( round( statistics.stdev(dat), 1) )
sem = str( round( statistics.stdev(dat)/math.\
sqrt(len(dat)), 1) )
print('\nGegeben sei ein Stichprobe mit den')
print("Daten: " + ", ".join(map(str,dat)))
print("als Teilmenge einer unbekannten \
Grundgesamtheit." + "\n")
print("Die Stichprobe enthält " + str(len(dat)) \
+ " Datenpunkte.")
print("\nWir schätzen für die Grundgesamtheit:")
print("Arithmetischer Mittelwert = " + mwt)
print("Median (Zentralwert) = " + med)
print("Standardabweichung (aus der Probe) \
= " + ssd + "\n")
print("Standardfehler des (geschätzten) Mittel\
werts = " + sem + "\n")
```

Mehr Module für statistische Berechnungen bietet die Python-Erweiterung Numpy.

QR-Code schreiben und lesen mit qrtools

QR-Code und das Programm `qrencode` zum Schreiben von QR-Code wurden bereits vorgestellt, siehe Seite 226. Das Python-Modul `qrtools` kann QR-Code schreiben und lesen (aus einer Bilddatei). Es wird durch

```
sudo apt install python3-qrtools
```

installiert. Folgendes Programm schreibt eine Bilddatei mit QR-Code.

```
1 # Program qrw.py writing QR code to a file
2 from qrtools import QR
3 from shutil import move
4 txt = "Wo viel Licht ist, ist auch Schatten."
5 qrc = QR(data=txt, pixel_size=10)
6 qrc.encode()
7 move(qrc.filename,"qrcode_message.png")
```

In Zeile 7 verschiebt die Methode `shutil.move` die Bilddatei, welche von `qrc.encode()` zunächst im Verzeichnis `/tmp` erzeugt wird, in das Arbeitsverzeichnis und benennt sie `qrcode_message.png`. Sie kann mit einem Bildanzeigeprogramm, zum Beispiel `fbi` oder `fim`, auf dem Bildschirm angezeigt werden (siehe Abschnitt 6.1.1 auf Seite 173).

Folgendes Programm liest eine Bilddatei mit QR-Code.

```
1 # Program qrr.py reading QR code from a file
2 from qrtools import QR
3 src = "qrcode_message.png"
4 qrc = QR(filename=src) ; qrc.decode()
5 txt = qrc.data ; typ = qrc.data_type
6 print("\ndatatype:", typ, "\n\n"+txt+"\n")
```

In Zeile 6 werden der „Datentyp“ der Zeichenkette `qrc.data` und `qrc.data` ausgegeben. Beginnt `qrc.data` mit `http://` oder `https://`, ist ihr „Datentyp“ `"url"`. Entsprechend werden noch weitere „Datentypen“ definiert. Liegt kein anderer „Datentyp“ vor, ist der „Datentyp“ `"text"` und es ist normaler Text. Abhängig vom „Datentyp“ kann man weitere Aktionen auslösen. Beim „Datentyp“ `"url"` könnte man zum Beispiel einen Browser mit `qrc.data` als URL aufrufen.

Graphen (im Sinne der Graphentheorie) zeichnen mit graphviz

Graphviz (Graphic Visualization Software) ist ein Programmpaket zur Zeichnung von Graphen (im Sinne der Graphentheorie), das um 1990 (in der Firma AT&T Labs) entstand und weiter entwickelt wird. Es wird durch

```
sudo apt-get install graphviz
```

installiert. Eine deutsche Einführung findet man hier.

Mit dem Python-Modul `graphviz` kann man das Programmpaket Graphviz steuern. Das Python-Modul wird durch

```
pip3 install --user graphviz
```

installiert. Man kann damit ungerichtete Graphen (graphs) und gerichtete Graphen (directed graphs, digraphs) zeichnen.

Folgendes Programm zeichnet einen gerichteten Graphen (digraph) mit drei Knoten (vertices, nodes) und 3 Kanten (edges) und speichert das Ergebnis in einer Bilddatei.

```
1  # Programm graph_1.py für ein Graphen-Bild
2  import graphviz
3  G1 = graphviz.Digraph() # Alternative: Graph
4
5  # Knoten (nodes)
6  G1.node(name="A", label="Anton",
7  color="lavender",style="filled")
8
9  G1.node(name="B", label="Berta")
10
11 G1.node(name="C", label="Cäsar")
12
13 # Kanten (edges)
14 G1.edge(tail_name="A",head_name="B",
15 color="blue", fontcolor="blue", label="e1")
16
17 G1.edge(tail_name="A",head_name="C",
18 color="red", fontcolor="red", label="e2")
19
20 G1.edge(tail_name="B",head_name="C",
21 color="blue", fontcolor="blue", label="e3",
22 constraint="false") # ohne Einfluss auf Layout
23
24 # Info und Ausgabe eines PNG-Bilds
25 f = "graph_1.png" # Pfad zur Bild-Datei
26 print(G1.source)
27 G1.render(engine="dot", outfile=f)
```

Abbildung 8.2 zeigt das Ergebnis, das in der Bilddatei `graph_1.png` gespeichert wurde. Das Attribut `constraint` bestimmt, ob eine Kante bei der automatischen Positionierung der Knoten berücksichtigt wird (`constraint="true"`, Standardwert) oder nicht (`constraint="false"`). Wenn in Zeile 22 `constraint="true"` gesetzt wird oder die `constraint`-Angabe weggelassen wird (`"true"` ist Standardwert), dann ergibt sich das in Abbildung 8.3 gezeigte Bild.

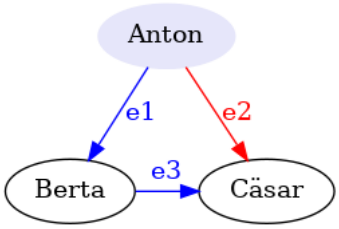


Abbildung 8.2: Ergebnis von `graph_1.py`

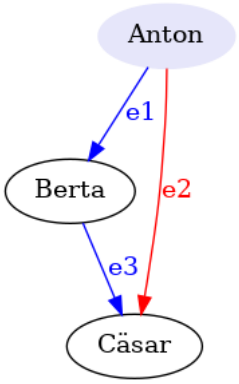


Abbildung 8.3: Ergebnis mit `constraint`

8.2.3 numpy, matplotlib.pyplot und scipy

numpy

Numpy ist eine umfangreiche Python-Bibliothek für schnelle Berechnungen mit großen Datenmengen. Man installiert Numpy durch

```
sudo apt install python3-numpy
```

Evenntuell muss eine weitere Bibliothek installiert werden.

```
sudo apt install libatlas-base-dev
```

Das Modul `numpy` wird üblicherweise unter dem Namen `np` importiert.

Wichtigste Datenstruktur in NumPy ist ein mehrdimensionales Array, das *ndarray* Objekt, nicht zu verwechseln mit dem `array.array` der Standardbibliothek von Python. Ein einfaches Array ist eine Zusammenfassung von Objekten des gleichen Datentyps. Ein strukturiertes Array enthält in jeder Spalte den gleichen Datentyp, aber für verschiedene Spalten können verschiedene Datentypen festgelegt werden. Die Größe des Arrays (Anzahl der enthaltenen Objekte) muss bei seiner Erzeugung festgelegt werden. Allerdings erlaubt NumPy auch ein Array von Objekten, welche zwar den gleichen Typ haben, doch unterschiedlich groß sein können, aber das verschlechtert die schnelle Verarbeitung, welche man mit NumPy erreichen will. Die genannten Eigenschaften unterscheiden ein Array von einer Liste in Python.

NumPy stellt viele Funktionen bereit, die schnell große Datenmengen in Arrays verarbeiten, mathematisch ausgedrückt: Abbildungen großer Mengen ähnlicher Elemente. Diese Funktionen sind maschinennah (in C) programmiert und daher viel schneller als die flexibleren Pythonfunktionen. Außerdem brauchen Daten in einem Array weniger Speicherplatz als in einer flexibleren Liste. Viele andere Pakete, zum Beispiel SciPy und Mathplotlib, benutzen NumPy.

Durch die Funktion `array()` wird ein ndarray, nennen wir es `myarr`, erzeugt. Der Datentyp dieses Arrays (nicht der Elemente), den wir mit `type(myarr)` erfahren, ist

`numpy.ndarray`. Argument von `array()` kann eine Liste, ein Tupel oder ein anderes Array-artige Objekt sein.

Arrays können eine beliebige Dimension $n \in \mathbb{N}$ haben. 0-D Arrays enthalten einen Wert (in der Physik Skalar genannt), 1-D Arrays eine endliche Folge (in der Physik Vektor genannt). 1-D Arrays werden oft durch eine Liste als Argument von `array()` gebildet und werden standardmäßig, mit dem Befehl `print(myarr)`, wie eine Liste angezeigt. Dimensionen werden oft *axes* genannt. Die Längen der axes werden bei der Erzeugung des Arrays festgelegt. Das Produkt dieser Längen ergibt die Anzahl der Elemente im Array. Die Längen der axes bestimmen das *shape* des Arrays. Zum Beispiel gibt das shape im 2-D Array an, wieviel Zeilen und Spalten enthalten sind. Mit dem Befehl

```
print(X.shape)
```

gibt man das shape eines vorher definierten Arrays `X` aus.

Ein eindimensionales Array mit äquidistanten Werten (aufeinander folgende Werte haben den gleichen Abstand) in einem vorgegebenen Intervall kann mit den Funktionen `arange()` und `linspace()` erzeugt werden. Bei `arange()` werden Intervallgrenzen und Abstand festgelegt (siehe das Beispiel unten), bei `linspace()` dagegen Intervallgrenzen und Anzahl der Werte.

Der Datentyp, den alle Elemente eines Arrays haben sollen, kann von NumPy automatisch bei der Erzeugung ermittelt werden. Man kann den Datentyp aber auch selbst mit der Option `dtype` festlegen, zum Beispiel wird mit `myarr = np.array([1, 2, 3], dtype='i1')` ein Array mit drei Elementen gebildet, die den Datentyp 1 Byte integer haben. Andere Datentypen sind `'u1')` für 1 Byte unsigned integer (0 – 255), `'b')` für boolean (True oder False, Speicherplatz 1 Byte), `'f4')` für 4 Bytes float, `'c8')` für 8 Bytes complex (bestehend aus zwei 4 Bytes float). Natürlich kann man die Zahl der Bytes für die numerischen Datentypen ändern.Zeichenketten können als bytestrings im Array gespeichert werden. Dabei gibt es keine Information über das encoding (Zeichencodierung), sondern nur eine Folge von Bytes. Zum Beispiel ist `'S10')` der Datentyp für Byte-Folgen (Zeichenketten) aus jeweils 10 Bytes. Das Encoding muss vor (und nach) der Speicherung im Array für die Umwandlung vom (oder in) Text angegeben werden. Es gibt weitere Datentypen und mehrere Bezeichnungsweisen für die Datentypen. Neben `'u1')` bezeichnet auch `np.uint8` den Typ 1 Byte unsigned integer.

Folgendes Programm erzeugt zwei NumPy-Arrays (Ar und Fo), multipliziert Elemente mit gleichem Index und gibt das Ergebnis (Array Mu) aus.

```
1 ''' Beispielprogramm zu NumPy '''
2 import numpy as np
3 Li = [-0.5,1.2,3.6] # Liste (auch ein Tupel ist möglich)
4 Ar = np.array(Li)   # NumPy-Array (Datentyp automatisch)
5 Fo = np.arange(start=1,stop=4,step=1,dtype='i1') # Folge
6 Mu = Ar*Fo # Multiplikation, Elemente mit gleichem Index
7 print("NumPy-Array Mu =",Mu)   # Ausgabe, wie bei Listen
```

In der Funktion `arange` wird ein halboffenes Intervall [Startwert,Stopwert) festgelegt, das heißt: Die Werte beginnen mit dem Startwert (im Beispiel bei 1) und enden vor dem Stopwert (im Beispiel bei 3). Die Schrittweite ist im Beispiel 1. Daher enthält das Array Mu die Werte 1, 2 und 3. Ihr Datentyp wurde als `'i1'` festgelegt, welcher ganze Zahlen zwischen -128 und +127 wiedergibt. Die Multiplikation mit $*$ bewirkt bei 1-D Arrays der gleichen Größe (allgemeiner Arrays mit gleichem shape) eine Multiplikation der Elemente mit gleichem Index. Sie darf nicht mit anderen Multiplikationsarten verwechselt werden! Das Programm liefert die Ausgabe `NumPy-Array Mu = [ -0.5 2.4 10.8]`.

Ein Array mit einer bestimmten Anzahl von äquidistanten Werten in einem Intervall kann mit der Funktion `linspace()` erzeugt werden. Im Unterschied zur Funktion `arange()` wird bei `linspace()` die Anzahl der Werte vorgegeben und daraus ergibt sich dann der Abstand der Werte. Mit den Parametern `start` und `stop` werden Anfang und Ende des Intervalls gegeben, mit `num` die Anzahl der Elemente des Arrays (Standardwert ist 50). Der Wert des Parameters `endpoint` entscheidet, ob ein geschlossenes [...] (`True`, das ist der Standardwert) oder halboffenes [...) Intervall (`False`) vorliegt. Der Parameter `retstep` entscheidet, ob der Rückgabewert das Array ist (`False`, das ist der Standardwert) oder ein Paar (Tupel aus zwei Elementen) aus Array und Abstand der Werte im Array (`True`) als Gleitkommazahl (float). Der Datentyp des erzeugten Arrays kann mit dem Parameter `dtype` festgelegt werden (Standard ist float), siehe oben. Das Beispielprogramm

```
1 ''' Werte im Intervall mit linspace '''
2 import numpy as np
3 A,d = np.linspace(start=0,stop=1,num=3,\
4 endpoint=True,retstep=True,dtype='f')
5 print("A: ",A," mit Abstand ",d)
```

ergibt `A: [ 0. 0.5 1. ] mit Abstand 0.5`

Ein 1-D Array oder 2-D Array kann man mit der Funktion `savetext()` als Textdatei gespeichert und mit `loadtext()` wieder eingelesen werden. Notwendige Parameter von `savetext()` sind der Name der Datei und der Name des Arrays. Man kann als weiteren Parameter eine Zeichenkette angeben, die (im altmodischen Stil von Python 2) das Format bestimmt, zum Beispiel `fmt='%d'` für Ganzzahlen (integer), da Zahlen standardmäßig als Gleikommazahlen (float) gespeichert werden. Als Spaltentrennzeichen zwischen den Werten dient das Leerzeichen, wenn man nicht mit dem Parameter `delimiter` etwas anderes bestimmt (Zum Beispiel ein Komma mit `delimiter=','`). Wenn man das Komma als Spaltentrennzeichen benutzt (CSV Format), muss man den Namen des Arrays in eckige Klammern einschließen (also als Liste übergeben). Notwendiger Parameter von `loadtext()` ist der Name der Datei. Der Rückgabewert, ein Array, sollte einer Referenz zugewiesen werden. Wenn das Spaltentrennzeichen in der Datei nicht das Leerrzeichen ist, muss es mit dem Parameter `delimiter` angegeben werden. Zahlen erhalten im Array standardmäßig den Datentyp einer Gleitkommazahl (float). Für Ganzzahlen (integer) muss man den Parameter `dtype='i'` setzen. Das folgende Programm schreibt ein Array

A1 in eine CSV-Datei fc mit Ganzzahlen, liest sie wieder ein und bildet ein Array A2 mit Ganzzahlen (integers).

```
''' Dateien lesen und schreiben mit NumPy '''
import numpy as np
A1 = np.array([-1,3,8],dtype='i') # Integer-Array
print("A1: ",A1," mit dem Datentyp:",A1.dtype)
fc = "testdatei.csv" # comma-separated values CSV
np.savetxt(fc,[A1],fmt='%d',delimiter=',')
A2 = np.loadtxt(fc,delimiter=',',dtype='i')
print("A2: ",A2," mit dem Datentyp:",A2.dtype)
```

matplotlib.pyplot

Matplotlib ist eine umfangreiche Python-Bibliothek für mathematische Graphiken. Eine Alternative ist die Grafikbibliothek DISLIN (siehe Seite 105). Die Installation von Matplotlib wurde bereits auf Seite 78 empfohlen.

```
sudo apt install python3-mathplotlib
```

Eventuell muss man eine weitere Installation vornehmen:

```
sudo apt install libatlas-base-dev
```

Wir beschränken uns auf das Modul pyplot, welches eine prozedurale (weniger objektorientierte) Schnittstelle bereitstellt, in die sich Nutzer, die MATLAB[10] kennen, gut einarbeiten können. Eine (nicht empfohlene) Alternative ist pylab.[11]

Die aktive Konfigurationsdatei für matplotlib findet man mit folgendem Programm.

```
# Programm "pyplot_1" mit Python 3, AHG (2020)
import matplotlib
print('config file:',matplotlib.matplotlib_fname())
```

Matplotlib erzeugt für eine Zeichnung ein Objekt, figure genannt, welches verschiedene Zeichnungselemente enthält. Mit pyplot spricht man das figure Objekt meistens nicht explizit (unmittelbar) an, aber die Funktionen von pyplot greifen auf Methoden und

[10] MATLAB ist ein kommerzielles Programm der Firma MathWorks für numerische Berechnungen und deren graphische Darstellung. Mit Python wird es durch NumPy, Matplotlib und SciPy ersetzt.

[11] Im Unterschied zu pyplot, einem Modul innerhalb von matplotlib, ist pylab ein Modul, welches anstelle von numpy und matplotlib.pyplot installiert wird. pylab enthält die Funktionalität von numpy und matplotlib.pyplot, aber die Syntax ähnelt MATLAB noch mehr und soll Nutzer ansprechen, die bereits mit MATLAB gearbeitet haben. Da pylab damit nicht dem systematischen Aufbau von Python folgt, bevorzugen wir numpy und matplotlib.pyplot.

Eigenschaften von `figure` implizit (mittelbar) zu. Standardwerte (default Matplotlib values) sind in einem Dictionary `matplotlib.rcParams` gespeichert.[12]

Das folgende Beispiel plottet Datenpunkte in einem Liniendiagramm und speichert es in einer Bilddatei. Das Modul `matplotlib.pyplot` wird üblicherweise unter dem Namen `plt` importiert.

```
1  # Programm "pyplot_2" mit Python 3, AHG (2020)
2  import matplotlib.pyplot as plt
3  # Daten für x- und y-Koordinaten, und ihre Gestaltung
4  lx = [-2,-1,0,1,2]          # Liste der x-Koordinaten
5  ly = [4.8,2,1.5,5.5,3]      # Liste der y-Koordinaten
6  fmt='bo-' # Formatangabe für Datenpunkte und  Linie
7  # Gestaltung der Achsen und Erzeugung des Plots
8  plt.xlabel('unabhängige Variable') # x-Beschriftung
9  plt.ylabel('abhängige Variable')   # y-Beschriftung
10 plt.axis([-3,3,0,6]) # Achsen: xmin,xmax,ymin,ymax
11 plt.plot(lx,ly,fmt)  # Plot-Erzeugung, keine Ausgabe
12 # Speicherung des Plots als Bilddatei mit Metadaten
13 md={'Title':'Messung 1','Author':'Max Schott'}
14 plt.savefig(fname='plot1.png',metadata=md)
```

Abbildung 8.4 zeigt das Ergebnis.

Im Beispiel gibt es einen Datensatz mit fünf Punkten. Die x- und die y-Koordinaten der Datenpunkte werden in jeweils eine Liste geschrieben. Die Formatierung des Datensatzes, das heißt Farbe und Form der Punkte und eventuell einer sie verbindenden Linie, wird in einer Zeichenkette angegeben (im Beispiel mit der Referenz `fmt`).

Mit den Funktionen `xlabel()` und `ylabel()` werden Beschriftungen für die x-Achse und die y- Achse gesetzt. Die Zeichenketten dürfen Escape-Sequenzen für Sonderzeichen enthalten, siehe Tabelle 8.10 auf Seite 286. Die Längen der Achsen können durch die Funktion `axis()` festgelegt werden; lässt man sie weg, werden sie automatisch gesetzt, was auch gut ist. Es ist meistens schöner, wenn die Achsenlängen etwas größer sind als die Wertebereiche von den x- und y-Koordinaten.

Die Funktion `plot()` erzeugt einen Plot der Datenpunkte oder eine Kurve, macht ihn aber noch nicht sichtbar. Parameter sind die x-Koordinaten, die y-Koordinaten (jeweils als Python-Liste oder NumPy-Array) und eine Formatierungszeichenkette (im Beispiel mit der Referenz `fmt`). In letzterer können einzelne Zeichen oder Zeichenpaare mit der in Tabelle 8.16 aufgeführten Bedeutung enthalten sein.

Die Funktion `savefig()` speichert die Graphik in einer Datei. Der Pfad zur Datei wird als Wert des Parameters `fname` angegeben. Wird das Format der Bilddatei nicht durch einen zusätzlichen Parameter bestimmt, ergibt es sich aus der Dateiendung (im Beispiel: `.png`). Der Parameter `metadata` dient zur Speicherung von Metadaten zum Bild.

[12] `rc` („run commands“) ist eine Bezeichnung aus frühen UNIX-Sytemen (1960er Jahre) und wird heute von geschichtsbewussten Informatikern gern in Fachausdrücken für Startwerte verwendet.

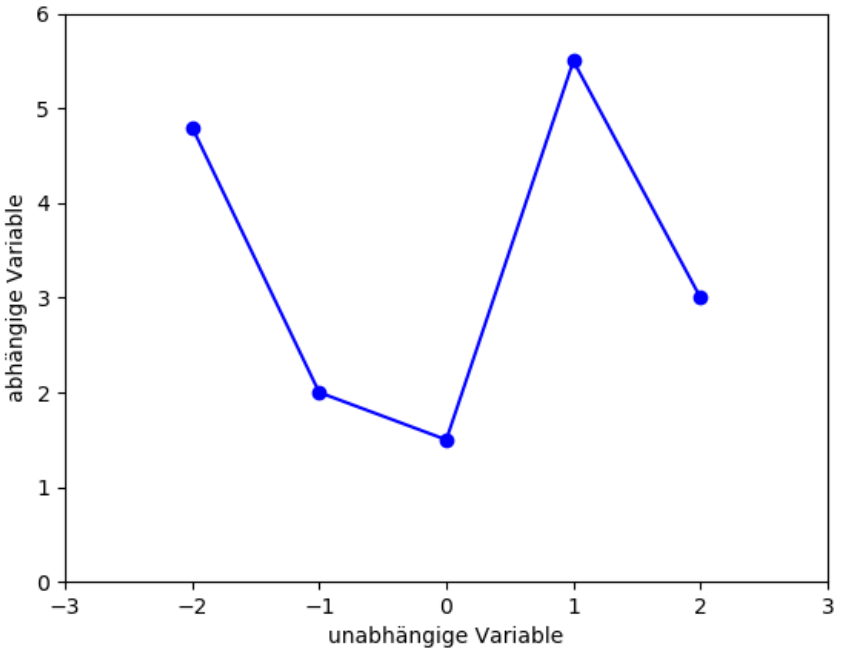


Abbildung 8.4: Liniendiagramm mit fünf Datenpunkten, in einer Bilddatei gespeichert

b	blau	o	gefüllter Kreismarker	-	durchgezogene Linie
g	grün	^	Dreiecksmarker ↑	--	gestrichelte Linie
r	rot	v	Dreiecksmarker ↓	-.	Strichpunkt-Linie
k	schwarz	s	quadratischer Marker	:	gepunktete Linie

Tabelle 8.16: Zeichen(paare) in der Formatzeichenkette von pyplot-Datensätzen

Eine PNG-Bilddatei kann, gemäß Regeln des World Wide Web Consortiums, Metadaten enthalten. Dabei werden bestimmten keywords (Schlüsseln) Zeichenketten als Werte zugewiesen. Man sollte nur die empfohlenen keywords verwenden, insbesondere: `Title`, `Author`, `Description`, `Copyright`. Angaben zur verwendeten Software (Matplotlib, einschließlich Versionsnummer), zum Erzeugungsdatum, und das Datum der letzten Änderung werden automatisch eingefügt. Dem Parameter `metadata` werden die Metadaten als Dictionary zugewiesen. Man kann natürlich auch darauf verzichten. Die Metadaten der PNG-Datei können mithilfe von ImageMagick gelesen werden, siehe Seite 182.

Die gespeicherte Bilddatei kann mit einem Bildanzeigeprogramm, zum Beispiel `fbi` oder `fim`, auf dem Bildschirm angezeigt werden (siehe Abschnitt 6.1.1 auf Seite 173).

Mittels PyGame kann man diese uns andere Matplotlib-Graphiken auch direkt (ohne Erzeugung einer Bilddatei) auf dem Bildschirm ausgeben (siehe Seite 396).

Das folgende Beispiel plottet eine Sinuskurve in einem Liniendiagramm und speichert

es in einer Bilddatei. Zur Darstellung werden Punkte der Sinuskurve berechnet und gezeichnet. Benachbarte Punkte werden mit einer Strecke verbunden. Damit man keine „Knicke“ in der Kurve sieht, muss die Anzahl der berechneten Punkte groß genug sein. Im Beispiel sind es 100. Die Berechnung der Koordinaten der Punkte der Sinuskurve erfolgt mit dem Modul `numpy` (siehe Seite 309).

```
# Programm "pyplot_3" mit Python 3, AHG (2020)
import numpy as np, matplotlib.pyplot as plt
# Daten für x- und y-Koordinaten, und ihre Gestaltung
lx,dx = np.linspace(start=-2*np.pi,stop=2*np.pi,\
                    num=100,endpoint=True,retstep=True)
ly = np.sin(lx) # y-Koordinaten = Sinus (x-Koordinaten)
print(" x-Werte-Abstand berechneter Kurvenpunkte:", dx)
fmt='g-' # keine Datenpunkte; grüne, durchgehende Linie
# Gestaltung der Achsen und Erzeugung des Plots
plt.xlabel('x') ; plt.ylabel('sin(x)') # Achsen-Beschr.
plt.axis([-2*np.pi,2*np.pi,-1.1,1.1]) # xmin,xmax,y ...
plt.plot(lx,ly,fmt)  # Plot-Erzeugung, keine Ausgabe
# Speicherung des Plots als Bilddatei mit Metadaten
md={'Title':'Sinusfunktion','Author':'Max Schott',\
'Description':'Beispiel','Copyright':'gemeinfrei'}
plt.savefig(fname='plot2.png',metadata=md)
```

Abbildung 8.5 zeigt das Ergebnis.

Die x- und y-Koordinaten wurden in jeweils einem eindimensionalen im Array gespeichert (im Beispiel `lx` und `ly` genannt, Datentyp `numpy.ndarray`).

Das Array der x-Werte wurden mit der Methode `linspace()` gebildet. Der Parameter `num` gibt die Anzahl der Werte vor. Sie liegen in einem Intervall, welches durch die Parameter `start` und `stop` begrenzt wird. Durch `endpoint=True` wird ein geschlossenes Intervall bestimmt (Alternative: halboffenes Intervall). Der Wert `True` für den Parameter `retstep` bewirkt, dass zusätzlich zum Array der x-Koordinaten der Abstand benachbarter x-Werte zurückgegeben wird (im Beispiel `dx` genannt).

Das Array der y-Werte wurde durch Anwendung der Sinusfunktion auf das Array der x-Werte gebildet. Die Zeichenkette zur Formatierung, im Beispiel mit der Referenz `fmt`, wählt eine grüne durchgehende Linie (vergleiche Tabelle 8.16).

In folgendem Beispiel werden zwei Kurven geplottet (`y1 = np.sin(x)` und `y2 = np.cos(x)`), jede mit eigener Formatierung, `fmt1` (durchgehende rote Linie) beziehungsweise `fmt2` ((durchgehende grüne Linie)). Es gibt verschiedene Beschriftungen und die Auflösung der gespeicherten Bilddatei (`plot3.png`) wird festgelegt.

```
# Programm "pyplot_4" mit Python 3, AHG (2020)
import numpy as np
import matplotlib.pyplot as plt
```

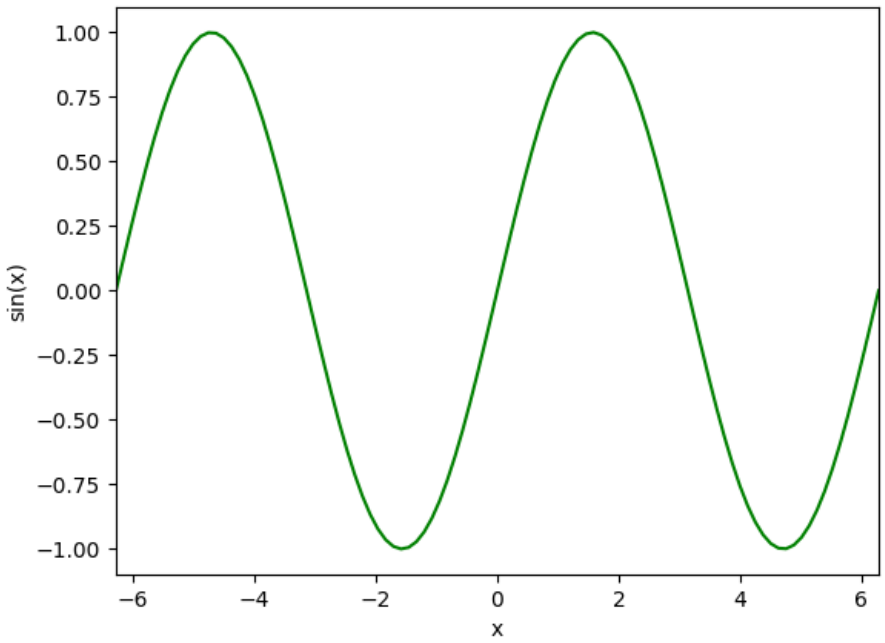


Abbildung 8.5: Sinusfunktion mit hundert Datenpunkten, in einer Bilddatei gespeichert

```
4  # Daten für x- und y-Koordinaten, und ihre Gestaltung
5  x = np.linspace(0,2*np.pi,100,endpoint=True)
6  y1 = np.sin(x) ; fmt1 = "-r" #   Sinus: rote Linie
7  y2 = np.cos(x) ; fmt2 = "-g" # Cosinus: grüne Linie
8  # Gestaltung der Achsen, Erzeugung des Plots, Legende
9  plt.xticks([0,np.pi,2*np.pi],['0',r'$\pi$',r'$2\pi$'])
10 plt.xlabel('x') # x-Beschr. (Legende statt y-Beschr.)
11 plt.axis([-0.1,2*np.pi+0.1,-1.1,1.1]) # xmin,xmax,y ...
12 plt.grid(c='b',linestyle='--',linewidth=0.5,alpha=0.1)
13 plt.plot(x,y1,fmt1,label='sin(x)')  # Label -> Legende
14 plt.plot(x,y2,fmt2,label='cos(x)')  # Label -> Legende
15 plt.legend(loc='lower left')    # Position der Legende
16 # Speicherung als Bilddatei mit bestimmter Auflösung
17 plt.savefig(fname='plot3.png',dpi=300) # dots per inch
```

Abbildung 8.6 zeigt das Ergebnis.

Während die Teilstriche der y-Achse automatisch gesetzt werden, gibt die Funktion `xticks()` die Lage und die Bezeichnung der x-Teilstriche (xticks) vor. Die Bezeichnungen können einfache Zeichenketten sein (`'0'`) oder aus LaTeX-Code gebildet werden (`$\pi$` und `$2\pi$`). Der LaTeX-Code steht in einem raw string (siehe Seite 287), damit der Backslash (\ ) nicht als Anfang einer Escape-Sequenz von Python (siehe Tabelle 8.10)

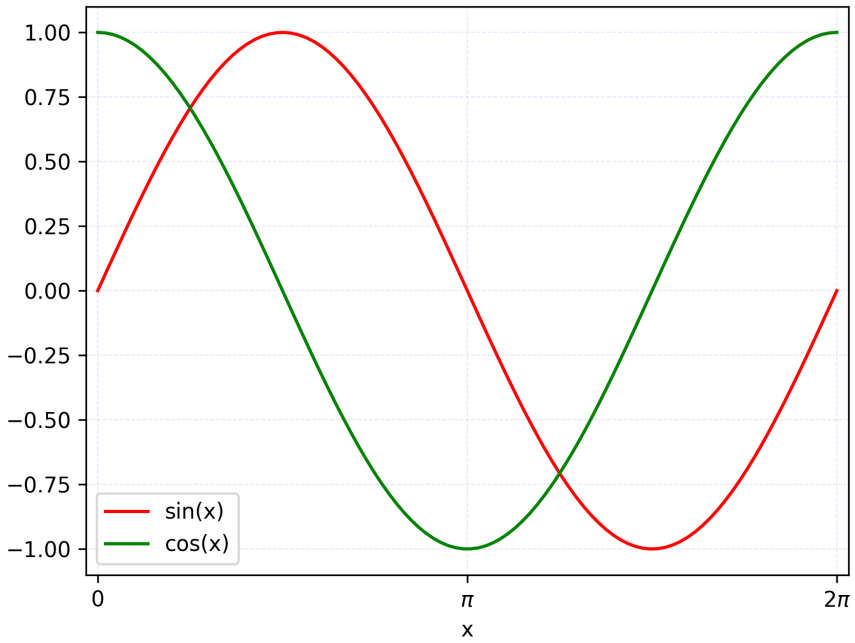


Abbildung 8.6: Sinus- und Cosinusfunktion, mit 300 dpi in einer Bilddatei gespeichert

gedeutet wird.

Die x-Achse wird mit der Funktion `xlabel()` beschriftet. Die y-Werte beider Kurven werden dagegen in einer Legende erklärt (siehe unten).

Die Funktion `grid()` erzeugt ein Gitternetz, das unter den Kurven liegt. Die Parameter `c`, kurz für `color`, und `linestyle` bestimmen Farbe und Stil der Gitterlinien (siehe Tabelle 8.16). Mit `linewidth` wird die Dicke der Linien festgelegt und `alpha` gibt den Wert für den alpha channel an, welcher die Transparenz gegenüber dem Hintergrund setzt (Standard: 1). 0 bedeutet maximale Transparenz, das heißt die Linien verschwinden, und 1 bedeutet minimale Transparenz, das heißt die Linien decken die Hintergrundfarbe ab. Ein Wert von 0,1 führt zu einem zarten, unaufdringlichen Gitternetz.

Die Funktion `plot()` erhält im Beispiel den Parameter `label`, welcher für jede Kurve eine Bezeichnung angibt, die in der Legende erscheinen soll. Die Funktion `legend()` erzeugt die Legende. Ihre Position im Plot wird mit dem Parameter `loc` bestimmt.

Bei der Speicherung der Datei mit der Funktion `savefig()` kann mit dem Parameter `dpi` die Auflösung (in der Einheit dpi) der Bilddatei festgelegt werden (Standard: 100). Der Parameter hat keinen Einfluss auf die Auflösung, mit der die Graphik auf einem Bildschirm gezeigt würde, wenn unsere Konsole das könnte. (In einem Linux-System mit X würde die Funktion `plt.show()` die Graphik auf dem Bildschirm zeigen.) Wenn man noch den Parameter `bbox_inches='tight'` hinzufügt, wird unnötiger weißer Rand abgeschnitten.

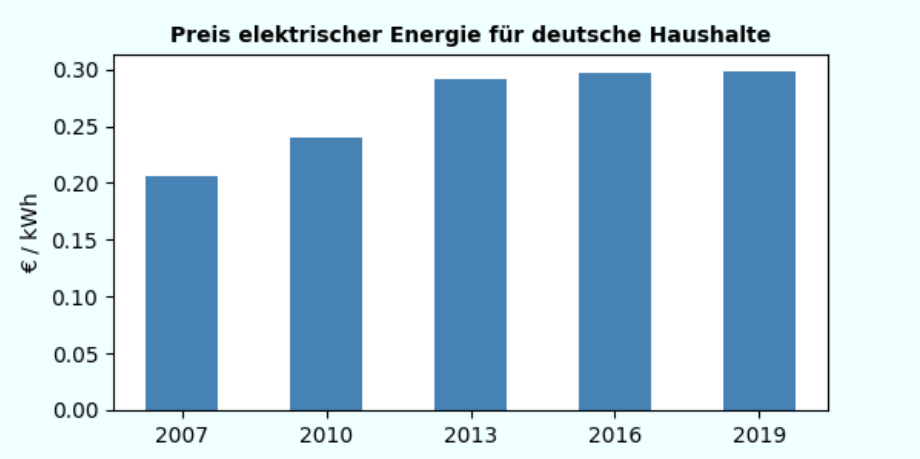


Abbildung 8.7: Säulendiagramm einer Zeitreihe des deutschen Strompreises

Folgendes Beispiel zeigt ein Säulendiagramm (vertical bar chart). Es hat einen formatierten Titel. Größe und Rahmenfarbe der Graphik wurden mit Parametern festgelegt.

```
1  # Programm "pyplot_5" mit Python 3, AHG (2020)
2  import matplotlib.pyplot as plt
3  dat = {"2007":0.2065, "2010":0.24065,
4  "2013":0.292, "2016":0.2973, "2019":0.2983}
5  plt.figure(figsize=(6,3), facecolor='azure')
6  fd={'fontsize':'medium','fontweight':'bold'}
7  plt.title('Preis elektrischer Energie für \
8  deutsche Haushalte',fontdict=fd,loc='center')
9  plt.ylabel('\u20ac / kWh') # mit Euro-Symbol
10 plt.bar(dat.keys(),dat.values(),
11         width=0.5,color="steelblue")
12 plt.savefig(fname='plot4.png')
```

Abbildung 8.7 zeigt das Ergebnis.

Die Daten sind in einem Dictionary abgelegt, deren Schlüssel (keys) Jahreszahlen sind (als Zeichenketten). Die Werte sind Gleitkommazahlen.

Die Funktion `figure` legt mit dem Parameter `figsize` Breite und Höhe der Graphik in der Längeneinheit Zoll (inch) fest. Der Wert ist ein Paar von Gleitkommazahlen, Standard ist (6.4,4.8). Der Parameter `facecolor` bestimmt die Rahmenfarbe.

Die Funktion `title` schreibt einen Titel. Der Parameter `fontdic` erhält ein Dictionary als Wert (im Beispiel mit der Referenz `fd`). Es dient der Formatierung der Titelschrift. Der Schlüssel `fontsize` legt die Zeichengröße fest. Mögliche Werte sind unter anderem `medium`, `large` (Standard), `x-large` und `xx-large`. Der Schlüssel `fontweight` bestimmt die Schriftstärke. Möglich sind unter anderem `normal` (Standard) und `bold` (fett). Der Parameter `loc` gibt die Position an, zum Beispiel `left` oder `center` (Standard).

Die Funktion `bar` erzeugt das Säulendiagramm, macht es aber noch nicht sichtbar. Die x-Koordinaten werden im Beispiel aus den Schlüsseln (keys) des Dictionary `dat` gebildet und die Säulenhöhen aus den Werten (values). Der Wert des Parameters `width` bestimmt die Breite jeder Säule (Standard: 0.8) und der Wer des Parameters `color` bestimmt die Farbe der Säulen. Die Funktion `savefig` speichert die Graphik in einer Bilddatei.

Lineare Regression mit SciPy

Die Python-Bibliothek `scipy` enthält Module für mathematische, natur- und ingenieurwissenschaftliche Berechnungen. Die Installation wurde bereits auf Seite 78 empfohlen.

Wir verwenden die Funktion `linregress` des Moduls `stats` von `scipy`, zusammen mit `numpy` (siehe Seite 309) und `matplotlib.pyplot` (siehe Seite 312), für die lineare Ausgleichsrechnung (lineare Regression).

In folgendem Beispiel werden fünf Datenpunkte (Messwerte) vorgegeben, deren x- und y-Koordinaten in jeweils einer Liste stehen. Es wird die Ausgleichsgerade $y = a + b \cdot x$ berechnet und deren Parameter ausgegeben, nämlich Steigung (slope) b und y-Achsenabschnitt (intercept) a. Außerdem werden verschiedene Größen, welche die Güte der Anpassung beschreiben, ausgegeben. Dann wird ein Plot der Messwerte und der Ausgleichsgeraden erzeugt und in einer Bilddatei gespeichert.

```
# Programm "pyplot_6" mit Python 3, AHG (2020)
from sys import exit
from scipy.stats import linregress
import matplotlib.pyplot as plt
from numpy import array
# Messwerte (je eine Liste mit x- und y-Koordinaten)
x = [ 0.6, 3.8, 5.5, 7.5, 9.9]    # x-Koordinaten
y = [-1.7, 1.1, 2.5, 3.8, 8.8]    # y-Koordinaten
if len(x) != len(y): exit(1) # Fehler -> Abbruch
# Berechnung der Ausgleichsgeraden y = a + b*x
b,a,r,p,e = linregress(x,y) # lineare Regression
print("Anzahl der Datenpunkte:",len(x))
print("Geradensteigung b (slope):",round(b,2))
print("y-Achsenabschnitt (intercept):",round(a,2))
print("Korrelationskoeffizient r:",round(r,2))
print("Bestimmtheitsmaß R^2 = r^2:",round(r*r,2))
print("p-Wert für Signifikanz von b:",round(p,2))
print("Standardfehler von b (stderr):",round(e,2))
# Plot der Messwerte und der Ausgleichsgeraden
fd={'fontsize':'medium','fontweight':'bold'}
plt.title(str(len(x))+' Messwerte, Ausgleichsgerade \
y = '+str(round(a,2))+' + '+str(round(b,2))+
' \u00b7 x, R\u00b2 = '+str(round(r*r,2)),fontdict=fd)
plt.xlabel('x') ; plt.ylabel('y')
```

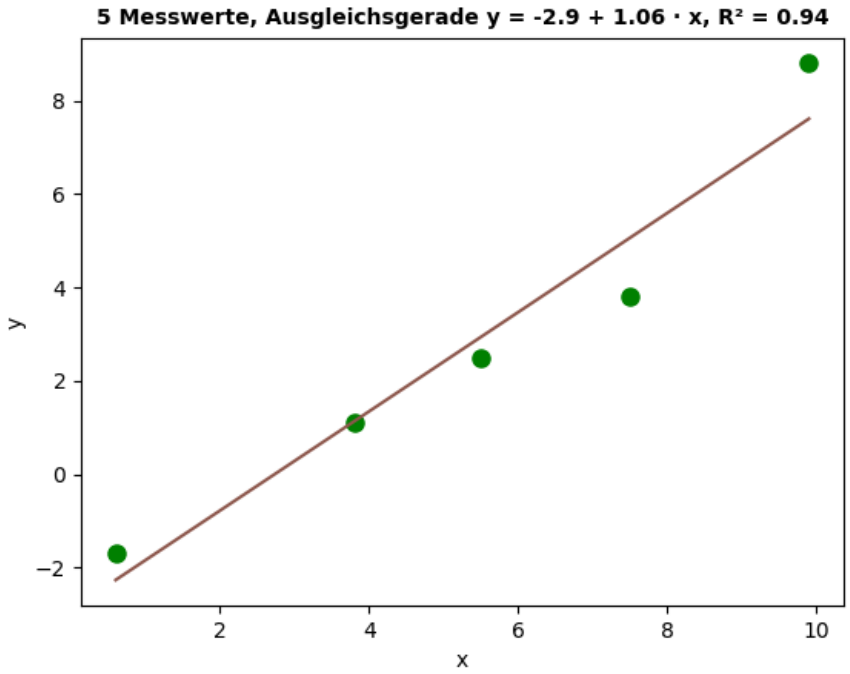


Abbildung 8.8: Einfache lineare Regression für fünf Messwerte (x_i, y_i)

```
25 plt.plot(x, y, 'o', color='g', markersize=8)
26 plt.plot(x, a+b*array(x), '-', color='tab:brown')
27 plt.savefig(fname='plot5.png')
```

Abbildung 8.8 zeigt das Ergebnis.

Die meisten Befehle wurden bereits früher erläutert. In der neunten Zeile des Programms wird geprüft, ob die Listen der x- und y-Koordinaten gleich viele Elemente enthalten. Falls nicht, wird das Programm über die Funktion `exit()` des Moduls `sys` mit dem Rückgabewert 1 (Fehler) verlassen.

Die beiden Escape-Sequenzen `\u00b7` und `\u00b2` im ersten Argument der Funktion `plt.title()` werden in Tabelle 8.10 auf Seite 286 erklärt.

Der Parameter `markersize` von `plt.plot()` gibt die Größe (Durchmesser in points, Standard: 6) der Marker (Symbole der Messpunkte) an. Entsprechend könnte man mit dem Parameter `linewidth` die Dicke der Linien einstellen (in points, Standard: 1.5).

Die `numpy`-Funktion `array` wandelt ein Array-ähnliches Objekt (im Beispiel eine Liste) in ein Array (Datentyp `numpy.ndarray`). Ein Array können wir, wie im Beispiel gezeigt, einfach mit einem Skalar multiplizieren oder zu einem Skalar addieren und erhalten wieder ein Array, welches als Argument für die Funktion `plt.plot()` taugt.

Nichtlineare Ausgleichsrechnung (curve fitting) mit SciPy

Im folgenden Beispiel verwenden wir die Funktion `curve_fit` des Moduls `optimize` der Python-Bibliothek `scipy`, sowie `numpy` (siehe Seite 309) und `matplotlib.pyplot` (siehe Seite 312), für die nichtlineare Ausgleichsrechnung (curve fitting, fit).

Im Beispiel werden fünf Datenpunkte (Messwerte) vorgegeben, deren x-Koordinaten und y-Koordinaten in jeweils einer Liste stehen. Die Listen heißen `x` und `y`. Gesucht wird eine mathematische Funktion, Modellfunktion genannt,

$$f : I = I_S \cdot \left(e^{-\frac{U}{n \cdot U_T}} - 1 \right) \tag{8.1}$$

deren Parametern I_S und n so gewählt sind, dass die Modellfunktion die Datenpunkte bestmöglich beschreibt. U_T ist, bei konstanter Temperatur T, eine Konstante.

„Bestmöglich beschreibt“ soll heißen, dass für die fünf Datenpunkte (x_i, y_i), mit $x_i \in$ `x` und $y_i \in$ `y` gilt

$$\sum_i \left(f(x_i) - y_i \right)^2 \quad \text{ist minimal} \tag{8.2}$$

Gleichung 8.1 heißt Shockley-Gleichung und beschreibt die Kennlinie einer Diode. Gleichung 8.2 heißt Summe der quadratischen Abweichungen. Wir benutzen Gleichung 8.2 in üblicher Weise und hinterfragen nicht, ob sie ein geeignetes Maß für die Anpassung der exponentiellen Modellfunktion ist.

```
1  # Programm "shockley" mit Python 3, AHG (2020)
2  from sys import exit
3  import numpy as np
4  from scipy.optimize import curve_fit
5  import matplotlib.pyplot as plt
6  from time import strftime
7  # Messwerte, Listen mit den x- und y-Koordinaten
8  d = "BAT43" # Name der Diode (für Beschriftung)
9  T = 25  # Temperatur in Grad Celsius
10 Ut = 1.380649*(T+273.15)/(10.0*1.602177)
11 x = [123.6, 167.5, 263.7, 288.7, 313.1] # mV
12 y = [7.9, 33.1, 512.2, 983.0, 2134.1] # uA
13 if len(x) != len(y): print("Fehler 1") ; exit(1)
14 # Bildschirm-Ausgabe
15 print("\nStart: Programm shockley,",strftime("\
16 %d. %B %Y, %H Uhr %M und %S s"))
17 # Definition der Modellfunktion f
18 def f(x,Is,n):
19     return Is*(np.exp(x/(n*Ut))-1)
20 # Fit der Modellfunktion an die Messwerte
21 p,c = curve_fit(f,x,y)  # Fit (ergibt Parameter)
22 mx = np.linspace(100,400,num=50)      # Modell-x
```

```
my = (p[0]*(np.exp(mx/(p[1]*Ut))-1)) # Modell-y
# Bildschirm-Ausgabe
print("Diode: ",d)
print("Temperatur T =",str(T),"\u00b0C,",\
"Temperaturspannung U_T =",round(Ut,3),"mV")
print("Shockley-Gleichung:\n      Saettigungs\
sperrstrom Is =",round(p[0],4),"\u03bcA","\n \
     Idealitaetsfaktor n =",round(p[1],3))
# Plot von Messwerten und Modellfunktion, Labels
plt.yscale('log') ; plt.grid(c='y',linewidth=0.2)
plt.plot(x, y, 'o', color='g', markersize=8,\
label='n = '+str(len(x))+' Punkte')
plt.plot(mx, my, '-', color='tab:brown',\
label="Shockley-Gleichung")
# Labels für drei theoretische Kennlinienpunkte
x10 = p[1]*Ut*np.log(10/p[0]+1) # U (I = 10 uA)
u10 = "U("+str(round(x10,1)).replace(".",",")+\
" V) = 10 \u03bcA"
plt.plot([],[],' ',label=u10) # Zeile in Legende
x100 = p[1]*Ut*np.log(100/p[0]+1) # U (I = 100 uA)
u100 = "U("+str(round(x100,1)).replace(".",",")+\
" V) = 100 \u03bcA"
plt.plot([],[],' ',label=u100) # Zeile in Legende
x1000 = p[1]*Ut*np.log(1000/p[0]+1) # U (I = 1 mA)
u1000 = "U("+str(round(x1000,1)).replace(".",",")+\
" V) = 1 mA"
plt.plot([],[],' ',label=u1000) # Zeile in Legende
# Titelzeile, Achsenbeschriftungen und Legende
l='$T$ = '+str(T)+' \u00b0C, $I$ = '+\
str(round(p[0],3)).replace(".",",")+\
' \u03bcA \u00b7 (exp('+' $U$ / ('+\
str(round(p[1],2)).replace(".",",")+\
' \u00b7 $U_T$) - 1))'+', $U_T$ = '+\
str(round(Ut,1)).replace(".",",")+' mV'
fd={'fontsize':'medium','fontweight':'normal'}
plt.title('Kennlinie einer Diode '+d+', '+\
strftime("%d. %B %Y, %H Uhr %M")+"\n"+l\
,fontdict=fd)
plt.xlabel('Spannung $U$  [mV]')
plt.ylabel('Strom $I$  [\u03bcA]')
plt.legend(loc='upper left')
# Speicherung der Graphik in einer Bilddatei
datei = strftime("fit_"+"%y%m%d%H%M%S"+".png")
md = {'Title':'Diodenkennlinie','Author':\
```

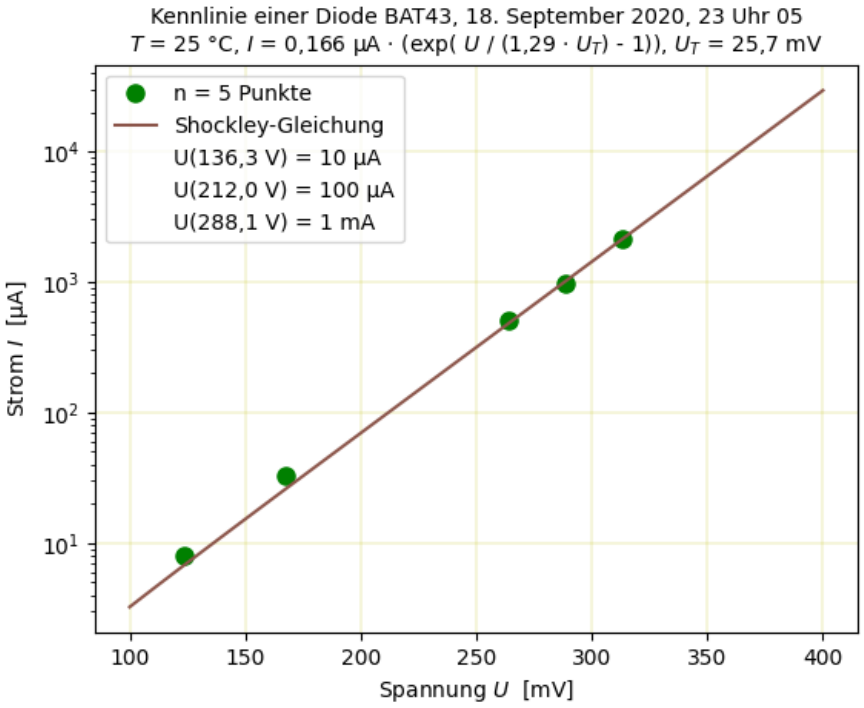


Abbildung 8.9: Kennlinie einer Diode, Messwerte und Fit der Shockley-Gleichung

```
67 'AHG','Description':'Shockley-Fit, Diode: '+d}
68 plt.savefig(fname=datei, dpi=100, metadata=md)
69 # Bildschirm-Ausgabe
70 print(" Stop: Programm shockley,",strftime("\
71 %d. %B %Y, %H Uhr %M und %S s")+"\n")
```

Abbildung 8.9 zeigt das Ergebnis.

Die Kurvenanpassung durch Ausgleichsrechnung wird kurz Fit genannt. Im Programm geschieht der Fit durch die Python-Funktion `optimize`, die als Argumente die Modellfunktion, sowie die Listen der Koordinaten der Datenpunkte erhält. Der Algorithmus für den Fit läuft automatisch: Der Nutzer muss das numerische Verfahren nicht steuern.

Rückgabewert ist ein Paar von Arrays (Datentyp jeweils `numpy.ndarray`). Das erste Array, `p`, enthält die bestmöglichen Werte der Parameter I_S (`p[0]`) und n (`p[1]`). Das zweite Array, `c`, enthält die Kovarianzen der Parameter. Wir betrachten es hier nicht.

Wissenschaftliche Konstanten von SciPy

Das Modul `constants` der Pythonbibliothek `scipy` stellt wissenschaftliche Konstanten und abgeleitete Einheiten bereit, wie das folgende Beispiel zeigt.

```
1 from scipy import constants
2 print("Astron. Einheit (astronomical unit)", end=" = ")
```

```
print(constants.astronomical_unit,"m")
print("Avogadro-Konstante (Avogadro const.)", end=" = ")
print(constants.Avogadro,"mol-1")
print("Boltzmann-Konstante (Boltzmann c.)", end=" = ")
print(constants.Boltzmann,"J/K")
print("Elektronenmasse (electron mass)", end=" = ")
print(constants.electron_mass,"kg")
print("Elektronvolt (electronvolt)", end=" = ")
print(constants.electron_volt,"J")
print("Elementarladung (elementary charge)", end=" = ")
print(constants.elementary_charge,"C")
print("Fallbeschleunigung (standard grav.)", end=" = ")
print(constants.g,"ms-2")
print("Feinstrukturkonstante (fine-str. c.)", end=" = ")
print(constants.fine_structure,"C")
print("Fuß (foot)", end=" = ")
print(constants.foot,"m")
print("Gallone, in den USA (gallon, US gal)", end=" = ")
print(constants.gallon_US,"m3")
print("Gaskonstante (universal gas const.)", end=" = ")
print(constants.gas_constant,"JK-1mol-1")
print("Gravitätskonstante (gravitat. c.)", end=" = ")
print(constants.gravitational_constant,"m3kg-1s-2")
print("Jahr (year)", end=" = ")
print(constants.year,"s")
print("Kalorie (calorie)", end=" = ")
print(constants.calorie,"J")
print("Knoten (knot)", end=" = ")
print(constants.knot,"m/s")
print("Planckssches Wirkungsq. (Planck c.)", end=" = ")
print(constants.Planck,"Js")
print("Lichtjahr (light-year)", end=" = ")
print(constants.light_year,"m")
print("Meile (mile)", end=" = ")
print(constants.mile,"m")
print("mmHg = Torr (torr = mmHg)", end=" = ")
print(constants.mmHg,"Pa")
print("Parsec (parsec)", end=" = ")
print(constants.parsec,"m")
print("Protonenmasse (proton mass)", end=" = ")
print(constants.proton_mass,"kg")
print("Rydberg-Konstante (Rydberg constant)", end=" = ")
print(constants.Rydberg,"m-1")
print("Seemeile (nautical mile)", end=" = ")
```

```
print(constants.nautical_mile,"m")
print("Stunde (hour)", end=" = ")
print(constants.hour,"s")
print("Tag (day)", end=" = ")
print(constants.day,"s")
print("Winkelgrad (degree of arc)", end=" = ")
print(constants.degree,"rad")
print("Winkelminute (minute of arc)", end=" = ")
print(constants.arcminute,"rad")
print("Zoll (inch)", end=" = ")
print(constants.inch,"m")
```

Mehr dazu findet man in der Dokumentation zu Konstanten von SciPy.

8.2.4 Audio, Video und Bilder

PyAudio

Das Modul `pyaudio` stellt eine Schnittstelle zur Bibliothek PortAudio bereit, welche zur Steuerung von Audioeingabe (recording) und Audioausgabe (playback) in verschiedenen Betriebssystemen eingesetzt werden kann.

Das folgende Beispielprogramm setzt voraus, dass ein Mikrophon über eine USB-Soundkarte angeschlossen ist. Es macht eine kurze Aufnahme und speichert diese in einer Datei im Waveform Audio File Format (WAVE, Dateiendung `.wav`).

```
# Program pa_record.py to record with USB microphone
import pyaudio, re, wave
from sys import exit

# Start of PyAudio
pa = pyaudio.PyAudio() # pyaudio.PyAudio instance

# Definition of variables
chk = 4096                  # 2^12 samples for buffer
fil = "test.wav"            # path to output audio file
frames = []                 # list of audio chunks
ndv = 0                     # number of USB Audio cards
re1 = re.compile(r"hw:\d,\d") # reg.expr. for ALSA cards
re2 = re.compile(r"USB.*Audio \(hw:(\d),\d") # reg.expr.
res = pyaudio.paInt16       # 16 bit resolution
sec = 3                     # recording time in seconds
dsr = "defaultSampleRate" # (16000, 44100, 48000)
spr = int(pa.get_device_info_by_index(2)[dsr])
chs = 1                     # number of channels
```

```
20
21 # Information to the user
22 pa = pyaudio.PyAudio() # pyaudio.PyAudio instance
23 print("\nIgnore the above messages from alsa-lib!")
24 print("\nALSA audio devices\n------------------")
25 for i in range(pa.get_device_count()):
26     dvi = pa.get_device_info_by_index(i).get('name')
27     if re1.search(dvi): print(dvi)
28     mch = re2.search(dvi)
29     if mch:
30         ndv = ndv + 1
31         dev = mch.group(1)
32 if ndv != 1:
33     pa.terminate()
34     exit("\nError in selecting USB sound card!\n")
35 print("\nWe will use ALSA sound card "+dev+".")
36 print("The sampling rate is", spr, "Hz.")
37 print("The duration of recording is", sec, "s.")
38
39 # Definition of audio stream
40 stream = pa.open(
41         format = res,
42         rate = spr,
43         channels = chs,
44         input_device_index = int(dev),
45         input = True,
46         frames_per_buffer = chk,
47 )
48
49 # Recording (append audio chunks to frame list)
50 n = int( (spr/chk) * sec )
51 t = str( round(n*chk/spr,1) )
52 input("Press ENTER to start recording!")
53 print("\nRECORDING", end=" ")
54 print("("+t+" s, "+str(n)+" chunks)", end=" ")
55 for i in range(0, n):
56      data = stream.read(chk,
57             exception_on_overflow = False)
58      frames.append(data)
59
60 # Stop of PyAudio
61 stream.stop_stream()
62 stream.close()
63 pa.terminate()
```

```
64 print("finished.")
65
66 # Save record to output audio file (WAV format)
67 out = wave.open(fil,"wb")
68 out.setnchannels(chs)
69 out.setsampwidth(pa.get_sample_size(res))
70 out.setframerate(spr)
71 out.writeframes(b''.join(frames))
72 out.close()
73 print("Audio file",fil,"written.\n")
```

Die gespeicherte Audiodatei kann mit einem Mediaplayer oder dem Programm `aplay` wiedergegeben werden (siehe Abschnitt 2.2.2 auf Seite 74).

```
aplay test.wav
```

Die Empfindlichkeit des Mikrophoneingangs und die Lautstärke der Wiedergabe kann man mit dem Programm `alsamixer` einstellen (siehe Abschnitt 2.2.1 auf Seite 67).

Audioaufnahmen durch ein Mikrophon überfordern oft den Raspberry Pi, insbesondere ältere Modelle. In obigem Programm sorgt die Option `exception_on_overflow = False` in Zeile 57 dafür, dass die Aufnahme trotz möglicher Überforderung nicht abbricht. Die Qualität der Aufnahme kann jedoch beeinträchtigt sein.

Man kann die Abtastrate (sampling rate) niedriger zu wählen, zum Beispiel 16 kHz statt 44,1 kHz, vorausgesetzt, dass das Mikrophon diese unterstützt. Das erleichtert dem Raspberry Pi die Aufnahme. Eine niedrigere Abtastrate verschlechtert zwar die Aufnahme von hohen Frequenzen, aber für Sprache kann es genügen.

Im Übrigen ist das Programm `arecord`, welches das Betriebssystem bereitstellt, oft einfacher und zuverlässiger als PyAudio (siehe Abschnitt 2.2.1 auf Seite 67).

vlc

Mit dem Modul `vlc` kann man den Mediaplayer VLC in einem Pythonprogramm steuern. VLC wird mit

```
sudo apt install vlc
```

installiert und das Modul `vlc` installiert man mit

```
pip3 install --user python-vlc
```

Folgendes Programm spielt einen Internet-Radiosender über den HDMI Audioausgang, wenn eine Internetverbindung besteht.

```
1 # Programm "vlc_radio.py" mit Python 3, AHG (2020)
2 import sys, vlc
3 from subprocess import run
```

```
4  #
5  # hoerspielprojekt.de (https://laut.fm/hoerspiel)
6  radio1 = "http://stream.laut.fm/hoerspiel"
7  # The UK 1940s Radio (http://1940sukradio.co.uk/)
8  radio2 = "http://1940sradio1.co.uk:8100/1"
9  # NHK World (https://www3.nhk.or.jp/nhkworld/en/live/)
10 tv1 = "http://nhkworld.webcdn.stream.ne.jp/www11\
11 /nhkworld-tv/global/2003458/live.m3u8"
12 # Radio Bremen Fernsehen (https://www.radiobremen.de/)
13 tv2 = "http://rbhlslive.akamaized.net/hls/live\
14 /2020435/rbfs/master.m3u8"
15 # ZDF neo (https://www.zdf.de/sender/zdfneo)
16 tv3 = "http://zdf-hls-16.akamaized.net/hls/live\
17 /2016499/de/high/master.m3u8"
18 #
19 # Lautstärke des Audio-Treibers (nicht VLC):
20 # ( amixer scontrols  zeigt Steuerungselement-Namen)
21 run(["amixer","-q", "sset","HDMI","100%"])
22 #
23 vi = vlc.Instance("--aout alsa --quiet") # vlc-Instanz
24 # --aout (audio output module) adummy, alsa, pulse
25 mp = vi.media_player_new() # MediaPlayer-Instanz
26 mp.audio_set_volume(100)   # VLC-Lautstärke 100%
27 #
28 m = vi.media_new(radio1)   # Medien-Instanz
29 mp.set_media(m)            # Medien -> MediaPlayer
30 # mp.audio_set_volume(50) # VLC-Lautstärke 50%
31 mp.play()                  # Start des Medien-Spiels
32 #
33 print(sys.argv[0],"Internet-Radio, Abbruch: Ctrl-C")
34 try:
35     while True: pass
36 except KeyboardInterrupt:
37     mp.stop()              # Wiedergabe ausschalten
38     sys.exit("\nAbbruch"+"\n")   # Programm beenden
```

Beendet wird die Wiedergabe durch die Tastenkombination `Ctrl-C` (siehe Seite 12).

Statt des Radiosenders, der in Zeile 6 mit der Referenz `radio1` gespeichert wurde, kann man auch einen anderen hören. Für den Radiosender, der in Zeile 8 mit der Referenz `radio2` gespeichert wurde, ändert man Zeile 28 in `m = vi.media_new(radio1)`. Auch die Wiedergabe lokal gespeicherter Audio-Dateien ist möglich.

Grundsätzlich kann man auch Internet-Fernsehen wiedergeben (siehe Zeilen 9 bis 17), aber der Erfolg hängt von der Leistungsfähigkeit des verwendeten Raspberry Pi Modells und von der Video-Auflösung des Datenstroms ab. Man muss etwas auf den Beginn der

Wiedergabe warten, insbesondere beim Sender, der oben unter `tv3` gespeichert ist. Zu beachten ist, dass sich die URL der Internet-Sender ändern können.

Normalerweise erfolgt die Wiedergabe von Videos im Vollbild. Der Befehl

```
mp.video_set_scale(0.6)
```

verkleinert das zentrierte Bild (andere Faktoren sind auch möglich), wenn `mp` eine Referenz auf die MediaPlayer-Instanz ist (wie in obigem Programm `vlc_radio.py`).

Für das Abspiel von Video-Dateien sollte man besser das `ncurses`-Modul benutzen, da die Wiedergabe von Videodateien mit dem Python-Modul `vlc` fehlerhaft ist.

pytube

Das Modul `pytube` ermöglicht das Herunterladen (Download) von YouTube Videos. Eine Alternative ist `youtube_dl` (siehe Seite 330). Man installiert `pytube` mit dem Befehl

```
pip3 install --user -U pytube
```

Mit der Option `-U` oder `--update` werden bereits vorhandene Pakete aktualisiert, wenn es eine neuere Version gibt. Folgendes Beispiel lädt und speichert ein YouTube Video.

```
1  # Programm pyt.py zum Download eines YouTube Videos
2  from pytube import YouTube
3  import os
4  #
5  yid = "WUvTyaaNkzM" # YouTube Video ID
6  nam = "EssenceOfCalculus-1_3Blue1Brown.mp4"
7  #
8  vdr ="/home/ahg/video/" # directory of videos
9  url = "https://youtube.com/watch?v="+yid
10 yto = YouTube(url) # YouTube object
11 print(yto.title,end=" ")  # title of video
12 print("("+str(yto.length)+" s)") # duration
13 vidstr = yto.streams.filter(progressive=True,
14 file_extension="mp4").get_lowest_resolution()
15 old=vidstr.download(vdr)
16 os.rename(old,vdr+nam)
```

In der Internet-Adresse (URL) eines YouTube Videos steht hinter der Zeichenkette `http://www.youtube.com/watch?v=` eine Zeichenkette, welche Video ID heißt und das Video eindeutig kennzeichnet. In Zeile 5 des Programms wird diese Video ID angegeben. In Zeile 6 wird der Dateiname angegeben, unter dem wir das Video speichern wollen. Die Zeilen 5 und 6 muss man anpassen, wenn ein anderes YouTube Video heruntergeladen werden soll. In Zeile 8 wird das Verzeichnis festgelegt, in dem Videodateien gespeichert werden. Die vollständige URL des YouTube Videos wird in Zeile 9 gebildet.

In Zeile 10 wird eine Instanz gebildet, die das YouTube Video repräsentiert. Sie hat unter anderem die Eigenschaften `title` und `length` für Titel und Dauer des Videos, welche in Zeilen 11 und 12 auf dem Bildschirm ausgegeben werden. In Zeilen 13 und 14 wird der Datenstrom bestimmt, der heruntergeladen werden soll. Mit dem Filter `progressive=True` wählen wir ein altes Format, in dem Audio und Video durch einen gemeinsamen Datenstrom transportiert werden. Die neuere Alternative (DASH) verwenden wir nicht. Der Filter `file_extension="mp4"` wählt das MP4-Format und die Methode `get_lowest_resolution` wählt den Videodatenstrom mit der geringsten Auflösung (resolution), der mit den Filtern vereinbar ist. Wir wählen hier die geringste Auflösung, weil wir unserem Raspberry Pi nicht zu viel zumuten wollen. In Zeile 15 wird der Datenstrom im in Zeile 8 festgelegten Verzeichnis unter einem Dateinamen mit der Referenz `old` gespeichert. In Zeile 16 wird die Datei umbenannt, wobei der in Zeile 6 angegebene Name verwendet wird. Das Video kann nun mit einem Mediaplayer, zum Beispiel VLC, abgespielt werden.

youtube_dl

Das Modul `youtube_dl` stellt in Python eine Schnittstelle zum Programm `youtube-dl` bereit, mit dem man Videodateien, zum Beispiel von YouTube, aus dem Internet herunterladen kann. Eine Alternative ist `pytube` (siehe Seite 329).

```
1  #!/usr/bin/python3
2  # Download einer Videodatei von YouTube
3
4  # URL (Datentyp str) mit Referenz url speichern
5  url = "http://www.youtube.com/watch?v=BaW_jenozKc"
6
7  # Optionen zum Download des Videos
8  # niedrigste Auflösung, um Speicherplatz zu sparen
9  ydl_opts = {"format": "worst"} # Datentyp dict
10
11 # Erzeugung einer Instanz ydl vom Datentyp YoutubeDL
12 from youtube_dl import YoutubeDL
13 ydl = YoutubeDL(ydl_opts)
14
15 # Hinzufügung von allen verfügbaren Extraktoren
16 ydl.add_default_info_extractors()
17
18 # Download und Erzeugung eines Dictionary info
19 with ydl:
20     info = ydl.extract_info(
21         url,
22         download=True # False: Info, kein Download
23     )
```

```
24
25 # Ausgabe des Namens der aktuellen Programmdatei
26 print ("\nAusgeführt wird Datei "+__file__)  # str
27
28 # Ausgabe des Titels des Videos (nicht mit IDLE)
29 # Der Titel kann aussergewöhnliche Zeichen enthaten.
30 print("Titel: " + info["title"])   # Datentyp str
31
32 # Ausgabe vom Namen des Uploaders und Upload-Datum
33 d = info["upload_date"][6:8]+". "+\
34 info["upload_date"][4:6]+". "+info["upload_date"][0:4]
35 print( "Upload von " + info["uploader"] + " ("+d+")" )
36 # + " (") +\ info["upload_date"] + ")")
37
38 # Ausgabe der URL der Webseite des Videos
39 print("URL: " + info['webpage_url'])   # Datentyp str
40
41 # Ausgabe der Bildgröße des Videos (Pixel)
42 print("Bildhöhe = " + str(info["height"]),\
43 ", Bildbreite = " + str(info["width"]))  # int, int
44
45 # Ausgabe weiterer Information zum Video
46 print("Länge = " + str(info["duration"]) + " s") # int
47 print("Untertitel = " + str(info["subtitles"])) # dict
```

Bildverarbeitung: PIL

Zur Bildbearbeitung in einem Pythonprogramm kann man ImageMagick (siehe Abschnitt 6.1.4 auf Seite 181) mithilfe der Methode `run` des Moduls `subprocess` verwenden (siehe Abschnitt 8.2.5 auf Seite 335), oder das Modul `PIL`.

Die Python Imaging Library PIL ermöglicht das laden, bearbeiten und speichern von Bilddateien und ermöglicht einfache Zeichnungen. Verschiedene Bildformate werden unterstützt, unter anderem PNG, JPEG, GIF, TIFF und BMP.

Die Bilder werden als Rastergrafik beschrieben, also als Rechteck, das aus kleinen farbigen Quadraten (Pixeln) besteht. Jeder Pixel hat, zusätzlich zur x- und y-Koordinate, eine Farbe. Bei der additiven Farbmischung wird jede vom Menschen wahrnehmbare Farbe durch die Überlagerung von drei Primärfarben (Rot, Grün und Blau) hergestellt. Das beruht auf den physiologischen Eigenschaften des Auges.

Im additiven Farbmodell wird die Farbe als Quadrupel `(R, G, B, A)` oder Tripel `(R, G, B` dargestellt.[13] R steht für den Rot-Kanal, G für den Grün-Kanal, B für den Blau-Kanal und A für den Alpha-Kanal. Bei einer Farbtiefe von 8 Bit ist jedes Element des Tupels eine ganze Zahl zwischen 0 und 255.

[13] Ein Quadrupel ist ein Tupel aus vier Elementen und ein Tripel ist ein Tupel aus drei Elementen.

Alle Farben lassen sich durch additive Farbmischung aus R, G und B herstellen. Soll ein Bild einen farbigen Hintergrund mehr oder weniger überdecken, kann man zusätzlich einen Alpha-Wert A angeben, welcher die Deckkraft beschreibt (Kehrwert: Transparenz). Mit einem Alpha-Wert von 255 wird der Hintergrund vollständig abgedeckt, während mit einem Alpha-Wert von 0 nur die Hintergrundfarbe erscheint. Beschränkt man sich auf Farben mit maximaler Deckkraft (Alpha-Wert = 255), kann die Angabe des Alpha-Werts entfallen. Tabelle 8.17 zeigt die RGB-Tupel einiger einfacher Farben.

Schwarz	Rot	Grün	Gelb	Weiß
(0, 0, 0)	(255, 0, 0)	(0, 255, 0)	(255, 255, 0)	(255, 255, 255)

Tabelle 8.17: RGB-Tripel einiger vollständig deckender Farben (ohne Alpha-Kanal

Wenn das Betriebssystem ein Bildanzeigeprogramm zur Verfügung stellt, können die Bilder auch gezeigt werden. In den folgenden Beispielprogrammen wird die Bilddatei vom Programm `fbi` (siehe Abschnitt 6.1.1 auf Seite 173) angezeigt. Dabei wird `fbi` durch die Methode `run` des Moduls `subprocess` aufgerufen (siehe Abschnitt 8.2.5 auf Seite 335).

Im ersten Beispiel zu PIL wird ein Bild verkleinert.

```
1 # Beispiel 1 zum Modul PIL
2 from PIL import Image # Bibl. PIL einbinden
3 from subprocess import run # run einbinden
4 b = Image.open("veronica.jpg") # Bilddatei
5 # print(b.size) # Bildpunkte je Zeile,Spalte
6 b_klein = b.resize((100,100)) # kleiner
7 b_klein.save("vkl.jpg") # Bild-Speicherung
8 run(["fbi","vkl.jpg"]) # zeigt kleines Bild
```

Im zweiten Beispiel zu PIL werden ein Rechteck und eine Linie gezeichnet.

```
1  # Beispiel 2 zum Modul PIL
2  from PIL import Image, ImageDraw   # Bibl. PIL einbinden
3  from subprocess import run # run einbinden -> fbi-Aufruf
4  #
5  mode = "RGB"    # Typ str, RGB 24 Bit (3x8 Bit) pro Pixel
6  size = (600, 400) # Typ tuple (int,int) hor.,vert. Pixel
7  color1 = (22, 27, 33) # Typ tuple 3*(0-255),"blackpearl"
8  color2 = (245,169,81) # Typ tuple 3*(0-255),"casablanca"
9  color3 = (255,0,0) # Typ tuple 3*(0-255),red: R255 G0 B0
10 # Instanz zur Definition der Bildfläche, Größe und Farbe
11 im = Image.new(mode, size, color1) # Typ PIL.Image.Image
12 # Instanz für die Zeichnung, Argument ist die Bildfläche
```

```
13 draw = ImageDraw.Draw(im)  # Typ PIL.ImageDraw.ImageDraw
14 #
15 # Rechteck, Lage: unten rechts, Größe 1/4 der Bildfläche
16 p1 = (300,200) # (x1,y1) Eckpunkt oben links (mittennah)
17 p2 = (599,399) # (x2,y2) Eckpunkt unten rechts (maximal)
18 draw.rectangle( [ p1,p2 ], fill=color2, outline=color3 )
19 # Streckenzug von links oben (schräg) bis fast zur Mitte
20 strecke = [(0,0),(299,199)] # 1., 2.Punkt (mehr möglich)
21 draw.line(strecke,fill=color3,width=12)  # width (Pixel)
22 #
23 # Speicherung, Bild in Datei, Format PNG ist verlustfrei
24 im.save("z.png") # Speicherung, im aktuellen Verzeichnis
25 run(["fbi","z.png"])  # Anzeige des Bild z.png durch fbi
```

imgkit (HTML → JPEG oder PNG)

Das Modul `imgkit` nutzt das Programm `wkhtmltoimage` (siehe Abschnitt 6.2.4 auf Seite 218), um HTML-Code oder eine Webseite in ein Bild zu verwandeln. Es wird mit

```
pip3 install --user imgkit
```

installiert und es setzt voraus, dass auch `wkhtmltoimage` installiert wurde.

Folgendes Beispiel gibt eine Webseite mit bekanntem URL in einer Bilddatei aus.

```
1 import imgkit
2 ziel = "https://de.wikipedia.org/wiki/Hugo_von_Jabala"
3 imgkit.from_url(ziel, "bilddatei.jpg")
```

Die Ausgabe kann entweder im Format JPEG oder im Format PNG erfolgen. Dies wird an der Endung der Bilddatei (`jpg` oder `png`) erkannt. JPEG komprimiert die Bilddaten und braucht weniger Speicherplatz. Die Methode `imgkit.from_file` nimmt statt eines URL den Pfad zu einer Datei als erstes Argument und `imgkit.from_string` nimmt entsprechend eine Zeichenkette (Datentyp `str`).

Die erzeugte Bilddatei kann mit `fbi` oder `fim` betrachtet werden (siehe Abschnitt 6.1.1 auf Seite 173). Die Darstellung (Rendering) einer Webseite erfolgt mit `imgkit` viel besser als mit den einfachen Browsern, die in Abschnitt 205 (Seite 6.2.1) vorgestellt wurden.

8.2.5 Betriebssystem, Python-Interpreter und neue Prozesse

os

Das Modul `os` stellt Schnittstellen zum Betriebssystem bereit. In folgendem Beispiel gibt die Methode `getpid` die Process ID `pid` des laufenden Prozesses (unseres Pythonprogramms) zurück und die Methode `getppid` liefert die `pid` des Elternprozesses.

```
1 import os
2 print( "Laufender Prozess:", os.getpid() )
3 print( "   Eltern-Prozess:", os.getppid() )
```

Wurde das Programm im normalen Terminal gestartet, ist der Elternprozess `bash`.

Folgendes Programm prüft, ob es die Datei `busch.txt` gibt. Gegebenenfalls wird sie gelöscht. Andernfalls gibt es keine Fehlermeldung.

```
1 # Programm "deldat" mit Python 3, AHG (2020)
2 import os
3 datei="busch.txt"
4 if os.path.exists(datei): os.remove(datei)
```

Folgendes Programm gibt die Werte einiger Umgebungsvariablen aus.

```
1 import os
2 print(os.getenv("BROWSER"))    # "w3m"
3 print(os.getenv("HOME"))       # "/home/ahg"
4 print(os.getenv("LANG"))       # "de_DE.UTF-8"
```

Die Kenntnis dieser Umgebungsvariablen hilft, wenn man aus dem Pythonprogramm einen Browser mit passenden Optionen aufrufen, eine bestimmte Datei im Heimatverzeichnis des Nutzers öffnen, oder eine Ausgabe in der Sprache des Nutzers machen will.

sys

Das Modul `sys` stellt Funktionen und Variablen des Python-Interpreters zur Verfügung.

Mit der Funktion `sys.exit` kann man ein Pythonprprogramm ordentlich beenden und zur Kommandozeile zurückkehren. Wird die Funktion ohne Argument aufgerufen, ist der Rückgabewert des Programms 0. Durch eine ganze Zahl > 0 kann man einen Fehler kennzeichnen. Der Rückgabewert kann von anderen Programmen gelesen werden, wenn diese das Pythonprogramm aufgerufen haben. In folgenem Programm

```
1 import sys
2 try:
3     with open("datei.txt","r") as f:
4             print("Datei lässt sich öffen.")
5 except:
6     print("Datei kann nicht geöffnet werden.")
7     sys.exit(42)
```

wird geprüft, ob sich die Datei `datei.txt` zum Lesen (texttt"r") öffnen lässt. Gegebenenfalls wird das in Zeile 4 mitgeteilt. Andernfalls wird der Ausnahme-Block (nach

except:) ausgeführt. In Zeile 6 wird der Fehler auf dem Bildschirm bekannt gegeben. Mit Zeile 7 verlässt man das Programm und das Argument 42 wird als Rückgabewert des Programms dem Betriebssystem übergeben. In der Kommandozeile kann man den Rückgabewert des zuletzt ausgeführten Befehls durch

```
echo $?
```

erfragen. Nach der Ausführung des obigen Pythonprogramms erhält man damit die Antwort 0 (wenn sich die Datei datei.txt öffnen ließ) oder 42 (wenn nicht).

Man kann ein Pythonprogramm mit Optionen (Kommandozeilenargumenten) aufrufen. Der Dateiname des Programms und die übergebenen Werte sind in der list sys.argv (argv steht für argument vector). In sys.argv[0] ist der Dateiname des Programms. Das erste Argument ist sys.argv[1], und so weiter. Beispiel:

```
1 import sys
2 print( "This is Programm", sys.argv[0] )
3 n = len(sys.argv)
4 if n==1:   x = "default value"
5 elif n==2: x = sys.argv[1]
6 else:
7     print( "Too many arguments!" )
8     sys.exit(1)
9 print( "String x is", x )
```

Die Länge von sys.argv ist um 1 größer als die Zahl der Argumente, da in sys.argv[0] der Dateiname steht. Alle Elemente von sys.argv haben den Datentyp str (string).

subprocess

Das Modul subprocess dient dazu, mithilfe des Betriebssystems neue Prozesse zu erzeugen, Eingaben dahin zu leiten und Ausgaben daraus aufzunehmen. Im Linuxsystem kann man damit Befehle an das Betriebssystem weiterleiten.

Soweit möglich, sollte man die Methode run benutzen, die in Python Version 3.5 eingeführt wurde. Erweiterte Möglichkeiten bietet die Methode Popen.

run erzeugt einen neuen (untergeordneten) Prozess (subprocess), welcher die im ersten Argument angegebenen Befehle ausführt und wartet, bis alle fertig sind. Beispiel:

```
1 # Programm "run" mit Python 3
2 from subprocess import run
3 x = [ "ls", "-l" ]
4 s = run( x, capture_output=True, text=True, check=True )
5 print(s.stdout)
```

Die Befehle, welche ausgeführt werden sollen, werden in der Regel in einer Liste (hier: x) oder einem Tupel als Argument an run übergeben. Die Elemente der Liste sind die durch Leerzeichen getrennten Teile der Befehlszeichenkette.

Die Option `capture_output=True` stellt (ab Python Version 3.7) die Rückgaben auf den Kanälen `stdout` und `stderr` zur Verfügung. Die Option `text=True` bewirkt (ab Python Version 3.7), dass die Ausgabe nicht mit dem Datentyp `bytes`, sondern mit dem Datentyp `str` erfolgt.

Wird die Option `check=True` nicht gesetzt (Standardwert ist `False`), läuft das Pythonprogramm weiter, wenn der subprocess mit einer Fehlermeldung (non-zero exit code) endet. Meistens ist es aber besser, `check=True` zu setzen. Dann wird bei einem fehlerhaften subprocess die Ausnahme (exception) `subprocess.CalledProcessError` geworfen und das Pythonprogramm abgebrochen.

Wenn die Option `shell=True` gesetzt ist (Standardwert ist `False`), werden die im ersten Argument angegebenen Befehle nicht direkt vom Betriebssystem ausführt, sondern an eine Shell zur Ausführung übergeben. Allerdings ist dies nicht die interaktive Login-bash, mit der wir normalerweise in der Konsole arbeiten (siehe Seite 108), sondern eine nicht-interaktive Non-Login-bash (siehe Seite 109). Mit `shell=True` können die angegebenen Befehle durch Metazeichen der `bash` verändert werden, was nützlich sein kann, aber es ist auch gefährlich.

Information zu Spannung und Temperatur der GPU

Elektrische Versorgungsspannung und Temperatur der GPU können mit dem Programm `vcgencmd` abgefragt werden, siehe Seite 126. Das wird in folgendem Programm benutzt, welches die Werte als Gleitkommazahlen bereitstellt und auf dem Bildschirm ausgibt.

```
1  # Programm "gpu_info" mit Python 3, AHG (2020)
2  from subprocess import run
3  from re import compile
4  reo = compile(r"\d+\.\d*")
5
6  cmd = "vcgencmd measure_volts".split()
7  ret = run(cmd,shell=False,check=True,\
8  capture_output=True,text=True)
9  erg=float(reo.search(ret.stdout).group())
10 print("SoC-Spannung:",erg,"V")
11
12 cmd = "vcgencmd measure_temp".split()
13 ret = run(cmd,shell=False,check=True,\
14 capture_output=True,text=True)
15 erg=float(reo.search(ret.stdout).group())
16 print("SoC-Temperatur:",erg,"\u00b0C")
```

Der Aufruf des Programms `vcgencmd` geschieht im Pythonprogramm mithilfe der Methode `run()` des Moduls `subprocess`. Diese erhält als Argument eine Liste (list) namens `cmd`, welche die einzelnen Worte des Befehls `vcgencmd measure_volts` beziehungsweise

`vcgencmd measure_temp` enthält. Die Methode `split()` wurde in Tabelle 8.11 auf Seite 288 vorgestellt.

Im Aufruf von `run()` setzen wir `shell=False`, da für die Ausführung des Programms `vcgencmd` die Shell (Bash) nicht gebraucht wird. Die Option `capture_output=True` stellt eine Ausgabe bereit und `text=True` wandelt den Rückgabewert des Befehls `vcgencmd` in eine Zeichenkette (str). Diese Zeichenkette erhalten wir als Eigenschaft `stdout` der Ausgabe `ret` von `run()`.

Die Zeichenkette enthält die gesuchte Zahl. Der Reguläre Ausdruck `reo` (siehe Abschnitt 8.1.7, Seite 290) beschreibt eine Gleitkommazahl (siehe Tabelle 8.12 auf Seite 291, `\.` steht für einen Punkt, keine Zeichenklasse). Die Methode `group()` (siehe Seite 291) gibt die extrahierte Gleitkommazahl als Zeichenkette (str) zurück und `float()` macht daraus den Datentyp Gleitkommazahl (siehe Seite 266).

Die Escape-Sequenz `\u00b0C` codiert das Gradzeichen (siehe Tabelle 8.10 auf Seite 286).

Bestimmung der internen IP-Adresse

Die IP-Adresse im lokalen Netzwerk kann mit dem Programm `hostname` und der Option `-I` abgefragt werden, siehe Seite 56. Dies wird in folgendem Programm benutzt.

```
1  # Programm "ip_adress" mit Python 3, AHG (2020)
2  from subprocess import run
3  cmd = "hostname -I".split()
4  ret = run(cmd,shell=False,check=True,\
5  capture_output=True,text=True)
6  try:
7      erg=ret.stdout.split()[0]
8      print("IP-Adresse:",erg)
9  except:
10     print("Kein Netzwerk!")
```

Die Verwendung der Methode `subprocess.run()`, sowie der Methode `split()` für Zeichenketten, wurde bereits für das Programm `gpu_info` auf Seite 336 erläutert. Aus der Zeichenkette, welche die IP-Adresse als erstes Wort enthält, macht die Methode `split()` eine Wortliste, aus der mit `[0]` das erste Wort (Index 0) entnommen wird.

Ist das lokale Netzwerk nicht erreichbar, gibt `hostname -I` nur ein Zeilenumbruchzeichen (newline), "\n") zurück und `ret.stdout.split()` ergibt eine leere Liste.

Zugriff auf das Dateisystem: shutil

Das Modul `shutil` erlaubt Operationen mit Dateien und Verzeichnissen. Da die Funktionen von `shutil` zum Kopieren (copy) nicht alle Metadaten (zum Beispiel den Eigentümer einer Datei) übertragen können, sollte man dafür die Funktionen des Betriebssystems mithilfe der Methode `run` des Moduls `subprocess` verwenden (siehe oben, Seite 335).

Mit der Funktion `which` kann man prüfen, ob ein Programm (ausführbare Datei) existiert und gegebenenfalls den Pfad zum Programm erfahren. Im folgenden Beispiel wird geprüft, ob es Programme mit dem Namen `mplayer` oder `mplaxer` gibt.

```
from shutil import which
l = ["mplayer", "mplaxer"]
for i in l:
    s = which(i)
    if s is None:
        print("Programm "+i+" gibt es nicht!")
    else:
        print("Pfad zum Programm "+i+": "+s)
```

Ist das Programm `mplayer` installiert, wird der Pfad zur Programmdatei ausgegeben. Da es ein Programm `mplaxer` nicht gibt, wird die entsprechende Meldung gezeigt.

8.2.6 Scanner steuern mit sane

Wenn ein Scanner angschlossen und SANE installiert wurde (siehe Seite 57), kann man das Einscannen mit einem Pythonprogramm steuern, nachdem man mit dem Befehl

```
sudo apt install python3-sane
```

das Modul `sane` installiert hat. Im folgenden Beispiel wird ein Dokument gescannt und in einer PDF-Datei im Unterverzeichnis `docum/scan/` des Heimatverzeichnisses des Nutzers gespeichert. Wenn der Dateiname nicht vom Nutzer eingegeben wird, dann wird er aus der aktuellen Zeit gebildet. Mithilfe des Moduls `pypdf`, installiert durch

```
sudo apt install python3-pypdf
```

kann in den Metadaten des PDF-Dokuments ein unsichtbarer Titel hinzugefügt werden, welchen man später zum Beispiel mit dem Programm `pdfinfo` auslesen kann (siehe Seite 179). Falls das Progrtamm `fbgs` installiert ist, wird die erzeugte PDF-Datei auf dem Bildschirm angezeigt (siehe Seite 179).

```
#!/usr/bin/env python3
# scan.py (AHG, 2023), output directory in line 25
# change value of my_name to your name in line 10
import os,sane
from subprocess import run
from string import ascii_letters
from time import strftime
from pypdf import PdfReader, PdfWriter
#
my_name = "Gitter"
```

```
11 san = sane.init()
12 if not sane.get_devices():
13     print("\nScan Program",__file__)
14     print("ERROR: no scanner found",\
15 "(use USB 3 or an active USB hub)\n")
16     sane.exit() ; os._exit(1)
17 else:
18     print("\nScan Program",__file__+\
19 ", scanner:",sane.get_devices()[0][2])
20 #
21 dev = sane.open(sane.get_devices()[0][0])
22 wh = dev.get_parameters()[2]
23 print("width x height (pixels):",str(wh[0]),"x",\
24 str(wh[1])+",","output file format: PDF")
25 dir = os.getenv("HOME")+"/docum/scan/" # directory
26 if os.path.isdir(dir)==False:
27     print("ERROR: missing directory", dir,"\n")
28     sane.exit() ; os._exit(1)
29 #
30 print("\nName your file without extension .pdf,")
31 dat = input("output file (ENTER for default): ")
32 dat = dat.strip().replace(" ","_") # remove spaces
33 if dat: # string not empty -> check file name
34     if dat[0] not in ascii_letters:
35         print("ERROR: must start with letter\n")
36         sane.exit() ; os._exit(1)
37     dat=dir+dat+".pdf" # path to output file
38     if os.path.exists(dat)==True:
39         print("ERROR: cannot overwrite file\n")
40         sane.exit() ; os._exit(1)
41 else: # string is empty -> use the default name
42     dat = dir+"scan"+strftime("%y%m%d%H%M%S.pdf")
43 #
44 tit = input("Title of PDF (ENTER for default): ")
45 tit = tit.strip().replace(" ","_") # remove spaces
46 if not tit:  # string is empty -> use the default name
47     tit = my_name+"_scan_"+strftime("%y%m%d")
48 #
49 # now we scan, save in PDF file and show result:
50 dev.start() ; img = dev.snap() ; img.save(dat)
51 dev.close() ; sane.exit()
52 rdr = PdfReader(dat) ; wrt =PdfWriter()
53 for p in rdr.pages: wrt.add_page(p)
54 wrt.add_metadata(rdr.metadata)
```

```
55 wrt.add_metadata({"/Title":tit})
56 with open(dat,"wb") as f: wrt.write(f)
57 try: run(["fbgs",dat]) # we need fbgs to show PDF
58 except: print("WARNING: cannot show output file\n")
59 print("\nEnd of scanning, output file:",dat,"\n")
```

Beim Dateinamen, den der Nutzer eingibt, wird darauf geachtet, dass er mit einem Buchstaben beginnt, und dass eine gleichnamige Datei im Verzeichnis noch nicht existiert.

8.2.7 HTTP requests

Kommunikation im World Wide Web

Die Kommunikation zwischen Programmen auf verschiedenen Rechnern geschieht häufig mit dem *Protokoll* (Regeln für den Informationsaustausch) *HTTP* (Hypertext Transfer Protocol; Versionen: HTTP/1.0, HTTP/1.1, HTTP/2, HTTP/3; HTTPS für verschlüsselte HTTP-Verbindung). Grundlage ist eine client/server architecture (client/server network), in der Daten über eine TCP/IP Verbindung gesendet werden.

Dabei unterscheidet HTTP das Programm, welches Dienste anbietet (*server*) von dem Programm, welches Dienste nachfragt (*client*).

Daten werden in Einheiten versendet, die als *messages* (Nachrichten) bezeichnet werden. Eine Anfrage (message vom client zum server) heißt request und die Antwort (message vom server zum client) heißt response. Anfragen beziehen sich auf ein Betriebsmittel, resource, über das der server verfügen kann, zum Beispiel eine HTML-Datei. Siehe Abbildung 8.10.

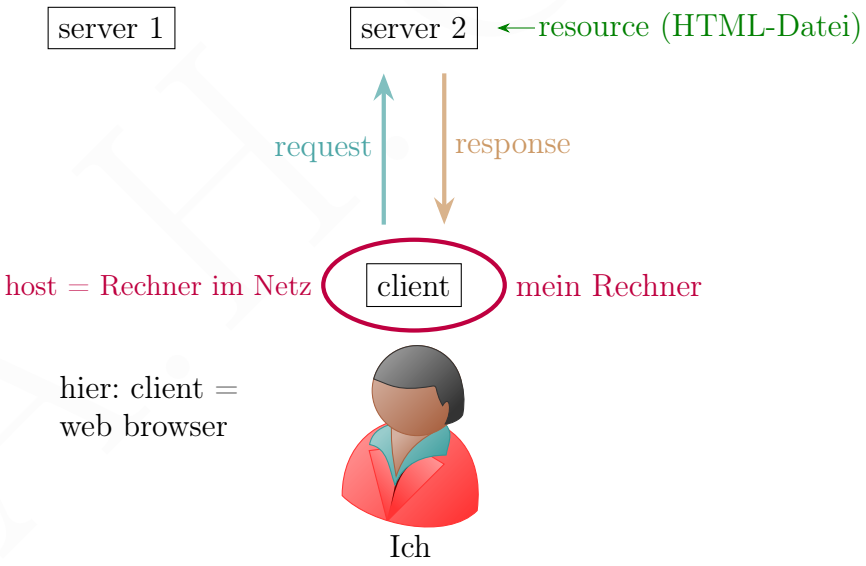


Abbildung 8.10: HTTP Kommunikation zwischen client und server

Im HTTP Protokoll hat jede message folgenden allgemeinen Aufbau:

1. start-line (request line oder response line) : eine Zeile, abgeschlossen mit CRLF
2. headers : beliebig viele header fields, jeweils abgeschlossen mit CRLF
3. CRLF, um das Ende der header fields zu markieren
4. message-body (optional)

wobei CRLF für Carriage Return (`\r`) und Line Feed (`\n`) steht.

Der Inhalt (payload) einer message, das heißt einige headers und (wenn vorhanden) der body, wird manchmal als *HTTP entity* bezeichnet. Nicht dazu gehören die start-line und formale headers.

HTTP requests (Anfragen) können nur mit bestimmten Methoden erfolgen, den request methods (siehe Tabelle 8.18). Die request methods (Anfragemethoden) wurden in RFC (Request for Comments) definiert. RFC ist eine Veröffentlichung von Regeln für das Internet. RFC 7231 beschreibt die request methods GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS und TRACE; es gibt eine Korrekturmitteilung dazu: https://www.rfc-editor.org/errata/rfc7231, und RFC 5789 fügt noch PATCH hinzu.

Jede resource (Betriebsmittel im Internet) hat mindestens einen Namen, Uniform Resource Identifier (URI) genannt. Ein URI darf nur ASCII-Zeichen enthalten. Wenn man diese Beschränkung aufhebt, spricht man von IRI (Internationalized Resource Identifier). Die Menge der URI enthält die Teilmengen der URL und URN. Die Begriffe URI, URL und URN werden kurz in RFC 3986 und ausführlich in RFC 3305 erläutert.

Der eindeutige Name einer resource, unabhängig von ihrem Ort im Internet, heißt URN (Uniform Resource Name). Eine Adresse, die einen Ort beschreibt, an dem eine resource zu finden ist, heißt URL (Uniform Resource Locator). Es ist möglich, dass eine resource in mehreren Kopien an verschiedenen Orten vorliegt. Daher kann sie mehrere URL haben. Oft genügt es, den Oberbegriff URI zu verwenden, statt zwischen URL und URN (und anderen URI-Typen) zu unterscheiden.

Ein URL wird für Webseiten (http) und andere resources, wie Dateiübertragung (ftp) und E-Mail (mailto), verwendet und hat einen bestimmten Aufbau.

`GET`	Anforderung einer resource (zum Beispiel eine Datei) vom server
`HEAD`	wie GET, aber ohne response body (response line und response headers)
`POST`	Senden großer Datenmengen (Beispiel: HTML-Formular) an den server
`PUT`	Senden einer Datenmenge an den server zum Ersatz einer resource
`DELETE`	Löschen einer bestimmten resource, auf die der server zugreifen kann

Tabelle 8.18: Wichtige HTTP request methods

Für HTTP-Anfragen, zum Beispiel zum Zugriff auf Dateien im Internet, ist das Modul `requests` geeignet. Ein request, der mit dem Modul `requests` erfolgreich durchgeführt wurde, gibt ein response Objekt zurück, welches über viele Attribute und Methoden verfügt, siehe Tabelle 8.18.

`r.headers`	liefert ein Dictionary der response headers
`r.close()`	schließt die Verbindung zum Server
`r.content`	liefert den Inhalt der response im Binärformat (bytes)
`r.text`	liefert den Inhalt der response als Text in Unicode
`r.status-code`	liefert den Statuswert der response als ganze Zahl
`r.reason`	liefert einen erklärenden Text zum Status der response
`r.ok`	liefert `True`, wenn der Statuswert < 200, sonst `False`

Tabelle 8.19: Attribute und Methoden des response Objekts `r` beim Modul `requests`

Die GET Methode von requests

Im ersten Beispiel zum Modul `requests` senden wir ein GET request an die Webseite von UniProt. UniProt ist eine frei zugängliche bioinformatische Datenbank für Proteine.

```
1 import requests
2 ziel = "https://www.uniprot.org/"
3 r = requests.get(ziel)
4 print(r.status_code)
```

Die Funktion `get` des Moduls `requests` erzeugt einen GET request, sendet ihn und gibt eine strukturierte Instanz zurück, die den response des servers darstellt. Sie erhält als Argument die URL einer resource (Zieladresse).

Die von `get` zurückgegebene Instanz ist vom Datentyp `requests.models.Response`. Sie hat ein Attribut (Eigenschaft) `status_code` vom Datentyp `int` (ganze Zahl), welche den dreistelligen Statuscode für den request angibt. Der Wert 200 für den Statuscode bedeutet, dass die Abfrage erfolgreich war (OK).

Beim Statuscode gibt es fünf Gruppen, entsprechen der ersten Ziffer des Statuscodes: 1, 2, 3, 4 oder 5. Statuscodes, die mit 1 beginnen, besagen, dass der request angekommen ist und (aus unterschiedlichen Gründen) noch bearbeitet wird. Eine 2 am Anfang des Statuscodes bedeutet, dass die Anfrage grundsätzlich erfolgreich ist. Ein Statuscode, der mit 1 beginnt, zeigt an, dass eine redirection (Umlenkung) der request notwendig ist. Wenn ein Fehler auftritt, der vom client verursacht wurde, beginnt der Statuscode mit 4. Bei einem Fehler, der vom server verursacht wurde, beginnt der Statuscode mit 5.

In folgendem Beispiel wollen wir die Aminosäuresequenz eines Proteins aus UniProt herunterladen.[14] Der Datenbankeintrag des gewünschten Proteins wird durch eine sogenannte accession number eindeutig gekennzeichnet. Hier ist sie P13569. Die Aminosäuresequenz soll als Zeichenkette in einem üblichen Format, FASTA, geholt werden.

[14] Genauer gesagt, ist es Isoform 1 (Standardform, canonical sequence) des CFTR-Proteins.

```
# Programm "uniprot_1", AHG (2020)
import requests, time
u1 = "https://www.uniprot.org/uniprot/"
u2 = "P13569"
u3 = ".fasta"
r = requests.get(u1+u2+u3, timeout=1)
time.sleep(1)
sf = r.text
print(r.url)
print(sf)
```

Die Funktion `get` des Moduls `requests` erhält hier als erstes Argument die URL und als zweites Argument die Zeitdauer in Sekunden, während der auf den Beginn der Antwort gewartet wird. Mit der Funktion `sleep` des Moduls `time` warten wir die angegebene Anzahl Sekunden, um sicherzustellen, dass die Antwort beendet ist, bevor wir die weitere Verarbeitung vornehmen. Die von `get` zurückgegebene Instanz `r` hat das Attribut `text`, welches den Textinhalt der Antwort als Zeichenkette (`str`) enthält. Das Attribut `url` gibt die URL der Anfrage als Zeichenkette zurück. Die Rückgabewerte von `r.url` und `r.text` werden auf dem Bildschirm ausgegeben.

Das FASTA Format beschreibt die Proteinsequenz durch eine mehrzeilige Zeichenkette. Die erste Zeile beginnt mit dem $>$ Zeichen und enthält einen kurzen Kommentar. Die Zeilen danach geben die Sequenz im einbuchstabigen Code für Aminosäuren wieder.

In der Diskussion des Erdklimas wird oft die Fläche des Meereises am Nordpol betrachtet. Die Veränderung der Meereisfläche während der letzten jahre wird in einer (täglich aktualisierten) Graphik des Danmarks Meteorologiske Institut (DMI) bildlich dargestellt. Folgendes Programm lädt die Bilddatei herunter und zeigt sie anschließend mit dem Bildanzeigeprogramm `fim`.

```
# Program sea_ice.py (AHG, 2022)
import requests
from subprocess import run

url = "http://ocean.dmi.dk/arctic/plots/\
icecover/osisaf_nh_iceextent_daily_5years_en.png"
filename = "arctic_ice.png"

response = requests.get(url)
if response.status_code == 200:
    with open(filename, "wb") as f:
        f.write(response.content)
else:
    print ('Error in HTTP request!')

```

```
16 run(["fim",filename], check=True)
```

Da die Bilddatei keine Textdatei ist, wird sie mit `wb` zum Schreiben einer Binärdatei geöffnet. Die Methode `run` des Moduls `subprocess` wurde ab Seite 335 vorgestellt. Das Bildanzeigeprogramm `fim` wird in Abschnitt 6.1.1 beschrieben (siehe Seite 173).

Auf der Webseite https://xkcd.com/ gibt es alle paar Tage einen neuen Cartoon. Das folgende Programm zeigt den neuesten.

```
1  #!/usr/bin/python3
2  # program ct.py shows latest cartoon from xkcd
3  import os, re, requests, subprocess
4
5  # dowload website text from server https://xkcd.com
6  try:
7      rsp = requests.get("https://xkcd.com")
8      txt = rsp.text
9  except requests.exceptions.RequestException as e:
10     raise SystemExit(e)
11
12 # search for image file name and save it in img
13 re1 = r'<img src="//imgs.xkcd\.com/comics/'
14 re2 = r'(\w+\.png)'
15 reo = re.compile(re1+re2)
16 mao = reo.search(txt)
17 if mao:
18     img = mao.group(1)
19 else: print("ERROR: image file not found!")
20
21 # dowload image file from server https://xkcd.com
22 bas = "https://imgs.xkcd.com/comics/"
23 try:
24     rsp = requests.get(bas+img)
25 except requests.exceptions.RequestException as e:
26     raise SystemExit(e)
27
28 # save image in temporary file, show it, delete it
29 with open("xkcd.tmp", "wb") as f:
30     f.write(rsp.content)
31 subprocess.run(["fim", "xkcd.tmp"], check=True)
32 os.remove("xkcd.tmp")
```

Das Herunterladen und Zeigen eines Bildes erfolgt ähnlich wie im vorherigen Beispielprogramm. Hinzu kommt hier die Auswertung des Textinhalts einer Webseite mithilfe

eines Regulären Ausdrucks (siehe Seite 290), um den Namen der Bilddatei mit dem aktuellen Cartoon zu finden. Ein ähnliches Beispiel, in dem ein Bild vom NASA Earth Observatory mithilfe von PyGame gezeigt wird, findet man auf Seite 395.

In folgendem Beispiel wird der Quellcode (HTML) von Webseiten mit Börsenkursen ausgewertet, die über eine Webseite der Tagesschau erreichbar sind. Es gibt je eine Webseite für den DAX, den MDAX und den TecDAX (Aktienindizes der Deutschen Börse AG). Sie werden heruntergeladen und mithilfe eines Regulären Ausdrucks (siehe Absschnitt 8.1.7 auf Seite 285) ausgewertet. Eine Liste mit Aktienkursen wird schließlich auf dem Bildschirm ausgegeben.

```
# Programm dax.py für Aktienkurse, AHG (2022)
import re, requests

def kurse(xax,url):
    req = requests.get(url, timeout=1)
    if req.status_code != 200:
        print("request error, code", req.status_code)
        print("URL:",url); exit()
    r1=r'<td>(.+?)</td>\s*<td class="ri"><span class="i'
    r2=r'con_\w{3,7}">(\d+,\d+)&euro;</span></td>\s*<td'
    r3=r' class="ri"><span class="\w{3,7}">(.+?)</span>'
    r4=r'</td>\s*<td class="ri"><span class="\w{3,7}">'
    r5=r'(.+?)</span></td>\s+.+?\s+<td class="ri hidden'
    r6=r'-xs-down">(.+?)</td>\s+<td class="ri hidden-xs'
    r7=r'-down hidden-sm-down">(.+?)</td>'
    reo = re.compile(r1+r2+r3+r4+r5+r6+r7) # re 6 groups
    ali = reo.findall(req.text)  # list of matching seq.
    n = 0      # initzialize output counter n
    for i in ali:
        if n%10==0: # check, if n is a multiple of 10
            print("{0:^25s}  {1:^9s}  {2:^7s} {3:^9s}\
  {4:^13s} {5:^9s}".format(xax+'-Aktie','Kurs','%',\
'\u0394','Zeit','Volumen'))
            print(80*"-")
        n=n+1 # increment output counter n
        print("{0:<25s}  {1:>9s}  {2:>7s} {3:>9s}  {4:\
>13s} {5:>9s}".format(i[0][0:25],i[1]+" \u20ac",\
i[2].replace("&plusmn;",""),i[3].replace(
"&plusmn;","")+" \u20ac",i[4],i[5]))
        if n%10==0: # check, if n is amultiple of 10
            x = input("\nFortsetzung: ENTER") ; print()

u1 = "https://www.tagesschau.de/"
u2 = "wirtschaft/boersenkurse/"
```

```
35 print("\nProgramm dax.py\n")
36 kurse("DAX",u1+u2+"dax-index-846900/")
37 kurse("MDAX",u1+u2+"mdax-index-846741/")
38 kurse("TecDAX",u1+u2+"tecdax-index-720327/")
```

In Zeilen 4 bis 31 wird die Funktion `kurse` definiert, welche die Parameter `xax` und `url` hat. Diese Parameter geben den Namen des Aktienindexes und die URL der zugehörigen Webseite an. In Zeilen 33 bis 38 stehen die URL und wird die Funktion `kurse` nacheinander für die Aktienindizes DAX, MDAX und TecDAX aufgerufen. Die Analyse einer Webseite und die formatierte Ausgabe der Aktienkurse erfolgten in der Funktion.

Innerhalb der Funktion `kurse` wird in Zeilen 5 bis 8 die Webseite geladen, oder im Fehlerfall das Programm vorzeitig beendet. Der Quelltext `req.text` der Webseite wird in Zeile 17 mit dem Regulären Ausdruck `reo` und der Methode `findall` analysiert. Das Ergebnis ist eine Liste, die für jede Aktie des Aktienindexes ein Tupel aus 6 Werten (Sextupel) enthält. Die Werte geben den Namen der Aktie, Kurs in Euro, prozentuale und absolute Veränderung, Zeit und Handelsvolumen an. In Zeilen 18 bis 31 wird für jede Aktie eine Zeile auf dem Bildschirm ausgegeben. Nach jeweils 10 Zeilen wird gewartet, bis er Nutzer durch Drücken der ENTER-Taste das Programm weiterlaufen lässt.

Der Reguläre Ausdruck `reo` wird in Zeilen 9 bis 16 definiert. Er bildet ein Zeichenkettenmuster (pattern), das für jede Aktie auf genau eine Zeichenkette (Sequenz) passt. Der Reguläre Ausdruck enthält 6 Gruppen; das sind Musterabschnitte, die durch runde Klammern eingefasst sind. Die 6 Gruppen passen auf 6 Teilsequenzen, welche die Werte des oben genannten Sextupels der Aktie sind (Namen der Aktie, Kurs in Euro, prozentuale und absolute Veränderung, Zeit und Handelsvolumen).

In folgendem Beispiel wird ein Landkartenteil, auf dem die EAH Jena zu sehen ist, von einer Webseite der OpenStreetMap Foundation geholt. Die Kartendaten, deren Urheber OpenStreetMap und seine Mitwirkenden sind, stehen unter der Open-Database-Lizenz. Die Lage der EAH Jena wird mit ihren geographischen Koordinaten angegeben. Das Kartenbild wird mithilfe des Moduls PIL (siehe Seite 331) gespeichert.

```
1  # Program osm_tile.py (AHG, 2022)
2  import requests
3  from math import pi, pow, log, radians, tan
4  from io import BytesIO
5  from PIL import Image
6
7  # Geographical coordinates of the EAH Jena
8  latitude, longitude = 50.918890, 11.569350
9
10 # Zoom factor (integer between 0 and 18)
11 zoom = 14
12
13 # Web Mercator projection and scale -> tile coordinates
```

```
14 z = pow(2, zoom) ; b = radians(latitude)
15 x = int( z * (longitude/360.0 + 0.5) )
16 y = int( (z/(2.0*pi)) * (pi-log(tan(pi/4.0+b/2.0))) )
17
18 # URL of OpenStreetMap tile showing the EAH Jena
19 osm = "https://tile.openstreetmap.org/"
20 url = osm+str(zoom)+"/"+str(x)+"/"+str(y)+".png"
21
22 # Save tile from OpenStreetMap to image file
23 with requests.get(url) as r:
24     img = Image.open(BytesIO(r.content))
25 img.save("geo.png")
```

In den Zeilen 14 bis 16 werden die Formeln der Web-Mercator-Projektion[15] angewandt, die bei kartographischen Webseiten üblich ist. Die durch Projektion entstandene ebene Karte der Erdoberfläche wird von kartographischen Webseiten in $2^{2 \times \text{zoom}}$ Teile zerlegt. Dabei ist zoom der Vergrößerungsfaktor. Ein solches Teil heißt englisch tile (deutsch Kachel) und ist, im Gegensatz zu einem Puzzleteil, quadratisch. Jede tile (Kachel) wird von kartographischen Webseiten digitalisiert und in einem Bild (zum Beispiel im Format PNG) der Größe 256 pixel × 256 pixel gespeichert. Das Bild kann vom Programm `fbi` (siehe Abschnitt 6.1.1 auf Seite 173) gezeigt werden. Die Bildgröße erfährt man, wenn `mediainfo` installiert wurde (siehe Seite 79), durch den Befehl

```
mediainfo geo.png
```

Setzt man zoom auf den kleinstmöglichen Wert 0, dann wird die Erdoberfläche auf nur einer tile abgebildet. Mit zoom = 18, dem größtmöglichen Wert in obigem Programm, wird die Erdoberfläche auf $2^{36} \approx 10^{11}$ tiles abgebildet.

Das obige Programm ermittelt anhand der geographischen Breite (englisch latitude) und Länge (longitude) eines Punkt der Erdoberfläche (hier: Standort der EAH Jena) bei gegebenem zoom die tile, welche den Punkt zeigt.

Wie ändern das Programm `osm_tile.py` nun, sodass nicht eine tile (Kachel) heruntergeladen wird, sondern ein zusammenhängendes Feld aus 7 × 5 tiles. Die tile, welche die vorgegebenen geographischen Koordinaten enthält, liegt in der Mitte. Das Feld der tiles wird zu einem Bild zusammengefasst und gespeichert. Am Ende wird das gespeicherte Bild mithilfe des Programms `fbi` auf dem Bildschirm gezeigt. Mit der Taste `q` beendet man das Programm und kehrt dann zur Kommandozeile zurück.

```
1 # Program osm_more.py (AHG, 2022)
2 import requests
```

[15] Die Web-Mercator-Projektion bildet die Oberfläche der Erde auf eine ebene Fläche ab, indem vereinfachend die Erde als ideale Kugel angesehen wird. Besonders bei großen Kartenmaßstäben (large-scale maps), also bei der Abbildung kleinerer Bereiche, ergibt sich eine fehlerhafte Abweichung von der genaueren Mercator-Projektion, welche die ellipsoide Form der Erde berücksichtigt.

```
3  from math import pi, pow, log, radians, tan
4  from io import BytesIO
5  from PIL import Image
6  from subprocess import run
7
8  # Variables defining the map of the EAH Jena
9  lat, lon = 50.918890, 11.569350 # geogr. coordinates
10 zoom = 18      # zoom factor (integer between 0 and 18)
11 nx = 3        # number of tiles to add east and west
12 ny = 2        # number of tiles to add north and south
13
14 # Web Mercator projection and scale -> tile coordinates
15 z = pow(2, zoom) ; b = radians(lat)
16 x = int( z * (lon/360.0 + 0.5) )
17 y = int( (z/(2.0*pi)) * (pi-log(tan(pi/4.0+b/2.0))) )
18
19 # Download and concatenation of tiles to form an image
20 kx = ky = 0   # indices for tile column and tile row
21 o = "https://tile.openstreetmap.org/" # OpenStreetMap
22 img = Image.new('RGB',((2*nx+1)*256,(2*ny+1)*256))
23 kx = 0       # reset the index for tile column
24 for i in range(x-nx,x+nx+1):
25     ky = 0   # reset the index for tile row
26     for j in range(y-ny,y+ny+1):
27         u = o+str(zoom)+"/"+str(i)+"/"+str(j)+".png"
28         with requests.get(u) as r:
29             tile = Image.open(BytesIO(r.content))
30         img.paste(tile,(kx*256,ky*256))
31         ky = ky + 1
32     kx = kx + 1
33
34 # Save and show image (end with key "q")
35 img.save("geo.png")           # save image
36 run(["fbi","-a","geo.png"])   # show image with autozoom
```

In Zeilen 9 bis 12 kann man (durch Angabe der geographischen Breite und Länge) den Ort wählen, der gezeigt wird, den Vergrößerungsfaktor zoom und die Anzahl der tiles, die neben der zentralen tile links, recht, oben und unten angefügt werden.

Die GET Methode mit Parameter-dictionary

Ein request kann die gewünschte resource durch Parameter genauer angeben. Die Werte dieser Parameter werden oft in einem Formularbereich einer Webseite eingetragen (der gemäß einer Formularvorlage = template gestaltet ist). Bei der GET Methode werden

solche Parameter als Teil der URL übertragen. Das HTTP Protokoll erlaubt zwar, dass ein GET request auch einen message-body enthält, aber üblicherweise fehlt er.

Nun soll ein request an einen server der Ernst-Abbe-Hochschule[16] (`eah-jena`) gesendet werden, um die aktuelle Belegung eines Raums, in dem Lehrveranstaltungen stattfinden können, zu erfragen. Die Parameter werden in einem dictionary (hier `d`) eingetragen.

```
1  # Programm "get-raum", AHG (2022)
2  from requests import Request
3  import imgkit
4  from subprocess import run
5  z1 = "https://www.eah-jena.de/"
6  z2 = "stundenplanung/raeume/detail"
7  d = {
8  "tx_eahstundenplanung_location[action]":
9  "detaillistdateweek",
10 "tx_eahstundenplanung_location[locationId]":
11 "D33590B4EA59CB23A68AF0FEA1B322BC", # Raum 02.02.01
12 }
13 p = Request("GET",z1+z2,params=d).prepare()
14 imgkit.from_url(p.url,"out.jpg")
15 run(["fbi","out.jpg"])
```

Das Modul `request` soll hier aber nicht den request absenden, sondern ihn nur vorbereiten. Die vorbereitete request-Instanz (hier: `p`) hat ein Attribut `url`, dessen Wert die URL der gewünschten resource ist, mit dem Datentyp `str` (Zeichenkette).

Diese URL wird hier einer Methode des Moduls `imgkit` übergeben, welche die request ausführt und die response in einer Bilddatei (`out.jpg`) speichert, siehe Abschnitt 8.2.4 auf Seite 333. Danach wird die Bilddatei vom Programm `fbi` (siehe Abschnitt 6.1.1 auf Seite 173) angezeigt. Dabei wird `fbi` durch die Methode `run` des Moduls `subprocess` aufgerufen (siehe Abschnitt 8.2.5 auf Seite 335).

Prüfung der Internetverbindung

Mit folgendem Programm wird geprüft, ob eine Internetverbindung besteht.

```
1  # ob_internet.py - AHG (2020)
2  import requests as r
3  try:
4      ro = r.get("http://www.google.com/", timeout=1)
5      print("Wir sind mit dem Internet verbunden.")
6  except (r.ConnectionError,r.Timeout) as exception:
7      exit("Fehler: Internetverbindung fehlt!")
8  print("Weiter geht's, im Internet!")
```

[16] Die EAH ist eine Hochschule für angewandte Wissenschaften (Fachhochschule) in Jena, Thüringen.

8.2.8 Weitere Internet-Dienste

Das Modul `imgkit` wurde bereits oben (im Abschnitt 8.2.4 auf Seite 333) beschrieben. Es kann unter anderem eine WebSeite als Bild in einer Datei speichern.

Information zur WLAN-Verbindung: wifi

Das Modul `wifi` wird leider nicht mehr gepflegt (unmaintained), funktioniert aber noch. Es erleichtert die Kontrolle über verfügbare WLAN Netzwerke. Wir gehen davon aus, dass ein virtual environment namens `mypy` geschaffen wurde (siehe Seite 262), gehen in das entsprechende Verzeichnis, aktivieren unser virtual environment und installieren das Modul `wifi` über `pip3`.

```
cd ~/progs/mypy
source bin/activate
python -m pip install wifi
```

Wir schreiben nun folgendes Programm

```
# mywlan.py - AHG (2022)
from wifi import Cell, Scheme
mycell = list(Cell.all("wlan0"))[0]
print("\nWLAN - Verbindung\n-----------------")
print("SSID =", mycell.ssid)
s = mycell.signal
t = "(unbrauchbar)"
if s >= -80: t = "(schlecht)"
if s >= -70: t = "(ausreichend ohne Video)"
if s >= -67: t = "(OK)"
if s >= -60: t = "(gut)"
if s >= -50: t = "(sehr gut)"
print("Signalstärke =", s, t)
print("Frequenz =", mycell.frequency)
print("Kanalnummer =", mycell.channel,"\n")
```

und erhalten grundlegende Information zu einer bestehenden WLAN- Verbindung durch

```
python mywlan.py
```

Mit dem Befehl

```
deactivate
```

beenden wir das virtual environment und kehren ins normale Betriebssystem zurück.

Eigenen Standort und öffentliche IP Adresse bestimmen

Wir gehen davon aus, dass ein virtual environment namens `mypy` geschaffen wurde (siehe Seite 262), gehen in das entsprechende Verzeichnis, aktivieren unser virtual environment

und installieren das Modul `ipinfo` über `pip3`.

```
cd ~/progs/mypy
source bin/activate
python -m pip install ipinfo
```

`ipinfo` ermöglicht es, die öffentliche IP-Adresse und den ungefähren Standort des Rechners (genauer: des Routers, welcher den Rechner mit dem Internetdienstanbieter verbindet) zu ermitteln, wie folgendes Beispiel in der Datei here.py zeigt.

```
1 # here.py - AHG (2022)
2 import ipinfo
3 # https://github.com/ipinfo/python
4 handler = ipinfo.getHandler()
5 details = handler.getDetails()
6 print("public ip:",details.ip)
7 print("     city:",details.city)
8 print(" latitude:",details.latitude)
9 print("longitude:",details.longitude)
```

Mit dem Befehl

```
python here.py
```

lassen wir das Programm laufen und mit dem Befehl

```
deactivate
```

beenden wir das virtual environment und kehren ins normale Betriebssystem zurück. Ein `bash`-Script, welches das Gleiche leistet, wurde bereits vorgestellt, siehe Seite 152.

Geographische Koordinaten zu gegebener Adresse

Zur Bestimmung der geographische Koordinaten (Breitengrad und Längengrad) einer gegebenen Adresse kann das Modul GeoPy dienen, welches mit

```
sudo apt install python3-geopy
```

installiert wird. Folgendes Programm zeigt ein Beispiel:

```
1 #!/usr/bin/env python3
2 # geokoo.py
3 from geopy.geocoders import Nominatim as nn
4 usr = "Raspberry Pi"
5 #
6 adr = "Greifgasse 3, Jena, Germany"
7 #
8 loc = nn(user_agent=usr).geocode(adr)
9 print("Adresse:", loc.address)
```

```
10 print("Breite:", loc.latitude)
11 print("Länge:", loc.longitude)
```

Webseite im Browser öffnen: webbrowser

Mit dem Modul `webbrowser` kann man aus einem Pythonprogramm einen Webbrowser aufrufen und eine Webseite (oder allgemein eine URL) darstellen lassen.

```
1 #!/usr/bin/env python3
2 # browse.py
3 import webbrowser, time
4 url = "https://de.wikipedia.org/wiki/Jena"
5 # brw = webbrowser.get("lynx")
6 brw = webbrowser.get("w3m")
7 brw.open(url)
8 print("Programm läuft nach Browserschließung weiter.")
```

Man kann auf diese Weise nur registrierte Browser öffnen. Dazu gehören `lynx` (siehe Seite 207) und `w3m` (siehe Seite 209), aber (noch) nicht `links2`.

Feedreader im Pythonprogramm: feedparser

Mit dem Modul `feedparser`, das mit dem Befehl

```
pip3 install --user feedparser
```

installiert wird, kann man einen Feedreader erzeugen:

```
1 #!/usr/bin/env python3
2 # program welt.py is a news aggregator (AHG (2020)
3 import feedparser
4 import time
5 url = "http://www.welt.de/?service=Rss"
6 f = feedparser.parse(url) # WebFeed holen
7 t = f['feed']['title'] # Titel des WebFeeds
8 with open("welt", "w") as w: # Textdatei welt
9     w.write("\n" + t + "\n"+"\n") # Feedtitel
10    for i in f.entries:
11        w.write(i.title+"\n") # Eintrags-Title
12        dp = i.published_parsed # Datumsobjekt
13        d = time.strftime("%d. %B %Y", dp) # Datum
14        d = d.replace(' 0',' ') # ohne führende 0
15        w.write(d+"\n") # Datum (Tag, Monat, Jahr)
16        w.write(i.summary+"\n") # Eintrags-Text
17        w.write(i.link+"\n") # Eintrags-Verweis
```

```
18          w.write("\n") # Leerzeile nach Eintrag
```

Damit wird ein Webfeed der Tageszeitung *Die Welt* in der Datei `welt` formatiert gespeichert. Man kann die Datei danach zum Beispiel mit dem Befehl `less welt` lesen.

Deutsches Fernsehprogramm

Eine Tagesübersicht des deutschen Fernsehrprogramms (mit Daten von HÖRZU) liefert das Pythonprogramm `tv_pr_tabelle.py` von Axel Schneider im Browser der Konsole. Unnötige Sender kann man im Dictionary `dictList` entfernen.

Aktienkurse mit yfinance

Das Modul `yfinance` wird mit dem Befehl

```
pip3 install --user yfinance
```

installiert. Es greift auf Börsendaten zu, die von *Yahoo Finance* bereitgestellt werden. Jede Aktie hat bei *Yahoo Finance* ein Symbol, was man auf der Webseite findet. Damit kann man den Kurs und andere Daten zur Aktie abfragen. Folgendes Programm erstellt eine Liste mit Börsenkursen.

```
1  # Programm boerse zur Anzeige von Börsenkursen
2  import yfinance as yf
3  a = {
4  "AFX.DE" : ["531370","Carl Zeiss Meditec   "],tstecdaxre
5  "SIE.DE" : ["723610","Siemens              "],
6  }
7  print()
8  print("{0:6s}  {1:21s}  {2:26s}  {3:22s}".\
9  format("WKN", "Aktie", "Kurs", "Bereich"))
10 print(79*"-")
11 for i in a:
12     b = yf.Ticker(i).info # Dictionary
13     p = "{0:6.2f}".\
14 format(b["regularMarketPrice"]).replace('.',',')
15     l = "{0:6.2f}".\
16 format(b["regularMarketDayLow"]).replace('.',',')
17     h = "{0:6.2f}".\
18 format(b["regularMarketDayHigh"]).replace('.',',')
19     c = b["currency"]
20     s = "{0:22s}".format(b["sector"])
21     print("{0:8s}{1:23s}{2:7s}{3:4s}({4:6s}-{5:6s}\
22 )  {6:22s}".format(a[i][0], a[i][1], p, c, l, h, s))
23 print()
```

Die Werpapierkennnummer (WKN) einer Aktie kann man in einer deutschen Börsen-Webseite nachschlagen, zum Beispiel bei der ARD Börseninformation. Einige Aktien-Daten findet man in Abschnitt 9.3 auf Seite 429.

Aktienkurse mit finanzen-fundamentals

Das Modul `finanzen-fundamentals` wird mit dem Befehl

```
pip3 install --user finanzen-fundamentals
```

installiert. Es greift auf Börsendaten zu, die von *Finanzen.net* bereitgestellt werden. Jede Aktie hat in diesem Modul einen besonderen Namen. Damit kann man den Kurs zur Aktie abfragen. Allerdings ist die Abfrage sehr langsam. Folgendes Programm gibt den Kurs der VW Vorzugsaktie an der Frankfurter Börse aus.

```
# Programm finanzen zur Anzeige von Börsenkursen
import finanzen_fundamentals.stocks as stocks
a = {
'volkswagen_vz-aktie' : 'VW Vz.', # WKN 766403
}
print()
print("{0:13s}{1:12s}{2:16s}".format\
("Aktie", "Kurs", "Zeit"))
print("-"*41)
for i in a.keys():
    b=stocks.get_price\
(stock=i,exchange="FSE",output="dict")
    p = "{0:6.2f}".format(b["Price"]).replace('.',',')
    c = b["Currency"]
    t = str(b["Timestamp"])[0:16]
    print("{0:13s}{1:7s}{2:5s}{3:16s}".format\
cd p    (a[i],p,c,t))
print()
#
# Erzeugung einer Datei aktie mit Info zu Aktien x
# d = stocks.search_stock("x", limit=3)
# with open("aktie","w") as f:
#     f.write(str(d))
```

Einige Daten für `finanzen-fundamentals` findet man in Abschnitt 9.3 auf Seite 429.

Eine Liste mit Aktienkursen des TecDAX wurde oben (siehe Seite 345) bereits mit den Modulen `re` und `requests` durch Analyse einer Webseite erzeugt.

Suchen mit Google

Das Modul `googlesearch` ermöglicht die Suche mit der Suchmaschine Google. Man installiert es mit dem Befehl

```
pip3 install --user google
```

Das Paket `googlesearch` benötigt das Paket `beautifulsoup4`. Folgendes Programm sucht mit Google und gibt maximal fünf Treffer auf dem Bildschirm aus.

```
1 from googlesearch import search
2 for x in search('"hair cell" cochlea',stop=5):
3     print(x)
```

Suchen und Abspielen von YouTube-Videos

Das Modul `scrapetube` ermöglicht die Suche nach YouTube-Videos. Das früher für diesen Zweck einsetzbare Modul `youtube-search-python` wird micht mehr gepflegt. `scrapetube` benutzt nicht die YouTube Data API. Man installiert es mit dem Befehl

```
pip3 install --user scrapetube
```

Folgendes Programm sucht YouTube-Videos und gibt deren Identifikationscode aus.

```
1 import scrapetube
2 vid = scrapetube.get_search("Wikinger")
3 for x in vid:
4     print(x['videoId'])
```

E-Mail: smtplib

Beim Versand einer E-Mail sendet ein email client[17] (E-Mail Programm des Nutzers) die Mail in der Regel zunächst an einen mail server[18] (Computerprogramm des E-Mail-Providers des Nutzers, welches die E-Mails verwaltet und weiterleitet), der sie dann zum Empfänger sendet.

Man benötigt ein E-Mail-Konto mit E-Mail-Adresse bei einem E-Mail-Provider wie GMX oder Gmail, weil E-Mails, die direkt von einem privaten Rechner gesandt werden, von empfangenden E-Mail-Anbietern oft nicht angenommen werden.

Für den Versand benutzt der email client in der Regel das Kommunikationsprotokoll SMTP (Simple Mail Transfer Protocol). Wenn der email client E-Mails vom mail server abholt, benutzt er das Kommunikationsprotokoll IMAP (Internet Message Access Protocol) oder das ältere POP3 (Post Office Protocol version 3).

[17] client = Programm, das Dienste im Netzwerk abfragt (Webbrowser, E-Mail-Programm, ...)

[18] Server = Programm, das Dienste im Netzwerk anbietet, zum Beispiel Web-Server (Protokoll: HTTP), Daten-Server im Internet (FTP, SFTP), Mail-Server im Internet (SMTP, POP3, IMAP) oder lokale Server im LAN (SMB, NFS); LAN = Local Area Network = örtliches Rechnernetz

Man kann das Python-Modul `smtplib` verwenden, um eine einfache Textnachricht (nur ASCII-Zeichen) als E-Mail zu versenden. Leider kann `smtplib` nur Passwörter mit ASCII-Zeichen verarbeiten. Passwörter, die andere Unicode-Zeichen enthalten, führen zu einer Fehlermeldung. Möglichkeiten, diese Schwachstelle zu umgehen kann man im Internet finden.

GMX Deutschland Das folgende Programm erfordert ein E-Mail-Konto beim E-Mail-Provider GMX (Deutschland). Der Zugriff für Programme des Nutzers muss freigegeben werden, wie bereits in Abschnitt 6.2.5 auf Seite 219 erläutert wurde.

Für das Kommunikationsprotokoll SMTP wird eine message `msg` in einem bestimmten Format zusammengesetzt. Das wäre auch mit einfachen Zeichenketten möglich, aber die Methode MIMEText erleichtert den Bau der message. Das Beispiel erlaubt nicht das Senden von Anhängen.

```
1  # Program ahgmail1.py for sending a text email
2  #
3  from email.mime.text import MIMEText
4  import smtplib
5  #
6  # email server data
7  SMTP_SERVER = "mail.gmx.net"
8  SMTP_PORT = 587
9  SMTP_USERNAME = "zxyu@gmx.de"
10 SMTP_PASSWORD = "password"
11 #
12 # message data
13 EMAIL_TO = ["alfred.gitter@eah-jena.de"]
14 EMAIL_LISTSEP = ", "
15 EMAIL_FROM = "zxyu@gmx.de"
16 EMAIL_SUBJECT = "Test"
17 EMAIL_TEXT="Python sendet diese E-Mail."
18 #
19 # output of info
20 print("\nSending Email via",SMTP_SERVER)
21 print("using email account",SMTP_USERNAME)
22 print("to",EMAIL_TO[0])
23 print("Text message:",EMAIL_TEXT)
24 #
25 # MIMEText object
26 msg = MIMEText(EMAIL_TEXT)
27 msg['Subject'] = EMAIL_SUBJECT
28 msg['To'] = EMAIL_LISTSEP.join(EMAIL_TO)
29 msg['From'] = EMAIL_FROM
30 #
```

```
31 # sending the mail
32 mail = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
33 mail.starttls() # Verschlüsselung
34 mail.login(SMTP_USERNAME, SMTP_PASSWORD)
35 mail.sendmail(EMAIL_FROM, EMAIL_TO, msg.as_string())
36 mail.quit()
```

Der Empfänger wird in diesem Programm nicht als einfache Zeichenkette angegeben, sondern als Element einer Liste von Empfängern. So können, in einer möglichen Erweiterung des Programms, mehrere Empfänger in einer Liste angeben werden. Die Methode `join`, deren Argument die Liste ist, fügt die Elemente der Liste zusammen. Die Zeichenkette `EMAIL_LISTSEP` ist die Instanz, für welche die Methode `join` aufgerufen wird. Sie wird bei der Zusammenfügung zwischen die einzelnen Elemente der Liste gesetzt. Damit wird `msg['To']` in obigem Programm schließlich eine Zeichenkette mit den Empfängern zugewiesen, in der die Empfänger durch Komma und Leerzeichen getrennt sind.

E-Mail mit attachment Das folgende Programm sendet eine E-Mail mit angefügter Textdatei (attachment) über ein E-Mail-Konto bei GMX (Deutschland).

```
1  # Program ahgmail2.py - email with attachment
2  #
3  from email.mime.multipart import MIMEMultipart
4  from email.mime.text import MIMEText
5  from email.mime.base import MIMEBase
6  from email import encoders
7  import smtplib
8  #
9  # email server data
10 SMTP_SERVER = "mail.gmx.net"
11 SMTP_PORT = 587
12 SMTP_USERNAME = "zxyu@gmx.de"
13 SMTP_PASSWORD = "password"
14 #
15 # message data
16 EMAIL_TO = ["alfred.gitter@eah-jena.de"]
17 EMAIL_LISTSEP = ", "
18 EMAIL_FROM = SMTP_USERNAME
19 EMAIL_SUBJECT = "Test"
20 EMAIL_TEXT="Python sendet diese E-Mail."
21 EMAIL_FILE="MeineDatei.txt" # attachment file
22 #
23 # output of info
24 print("\nSending Email via",SMTP_SERVER)
25 print("using email account",SMTP_USERNAME)
```

```
26 print("to",EMAIL_TO[0])
27 print("Text message:",EMAIL_TEXT)
28 print("Attachment:",EMAIL_FILE)
29 #
30 # MIMEMultipart object
31 msg = MIMEMultipart()
32 msg['From'] = EMAIL_FROM
33 msg['To'] = EMAIL_LISTSEP.join(EMAIL_TO)
34 msg['Subject'] = EMAIL_SUBJECT
35 msg.attach(MIMEText(EMAIL_TEXT, "plain"))
36 with open(EMAIL_FILE, "rb") as att:
37     bas = MIMEBase("application", "octet-stream")
38     bas.set_payload(att.read())
39 encoders.encode_base64(bas)
40 bas.add_header("Content-Disposition",\
41 "attachment", filename=EMAIL_FILE)
42 msg.attach(bas)
43 #
44 # sending the mail
45 mail = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
46 mail.starttls() # Verschlüsselung
47 mail.login(SMTP_USERNAME, SMTP_PASSWORD)
48 mail.sendmail(EMAIL_FROM, EMAIL_TO, msg.as_string())
49 mail.quit()
```

`EMAIL_FILE` gibt den Pfad zur anzufügenden Textdatei an (hier: `MeineDatei.txt`).

EAH Jena (Deutschland) Die Verbindungsdaten können natürlich angepasst werden. Zum Beispiel sind die Server-Daten für den E-Mail-Dienst der Ernst-Abbe-Hochschule Jena: `SMTP_SERVER = "smtp.eah-jena.de"` und `SMTP_PORT = 587`

Gmail von Google Die Server-Daten für Gmail, den E-Mail-Dienst von Google sind: `SMTP_SERVER = "smtp.gmail.com"` und `SMTP_PORT = 587`. Der Zugriff für Programme des Nutzers muss aber erst freigegeben werden. Dafür meldet man sich im Google-Konto an, wählt „Google-Konto verwalten“ und unter „Sicherheit“ erlaubt man den „Zugriff durch weniger sichere Apps“. Manchmal gibt es Nachfragen von Google zu den Sicherheitseinstellungen, die man aber leicht beantworten kann.

8.2.9 Ansteuerung der GPIO

Die Ansteuerung der GPIO in der Shell über `sysfs` des Linux Kernels wurde in Abschnitt 4.1.5 auf Seite 128 geschildert und die Ansteuerung mit `pigpio` auf Seite 129.

In Python gibt es für die Ansteuerung der GPIO zwei Module zur digitalen Ein- und Ausgabe über einzelne Pins, nämlich RPi.GPIO (Ben Croston, 2012) und GPIO Zero

(Ben Nuttall, 2015), sowie besondere Module zur Arbeit mit Datenbussen. In ähnlicher Weise kann man mit Hilfe des Programms *WiringPi* die Ein- und Ausgabe von Daten über die GPIO in C und C++-Programmen steuern.

Bei der Ansteuerung einzelner GPIO in Pythonprogrammen bevorzugen neue Tutorials GPIO Zero, da es für Anfänger einfacher ist, wenn man sich genau an die vorgegebenen Rezepte hält. Es gibt aber sehr viele bewährte Programme, in denen RPi.GPIO eingesetzt wurde und die als Vorlage für eigene Projekte dienen können.

GPIO Zero

`GPIO Zero` wird mit dem Befehl

```
sudo apt install python3-gpiozero
```

installiert. Die GPIO werden über die Nummer im BCM-Kanalnamen des System-on-a-Chip adressiert (siehe Seite 19); GPIO-23 also über die Zahl 23.

In folgendem Beispiel benutzen wir die in Abschnitt 1.2.4 auf Seite 32 beschriebene Schaltung. Eine Leuchtdiode ist über einen Vorwiderstand an GPIO 23 angeschlossen. Man kann auch die entsprechende Transistorschaltung nehmen, siehe Seite 34.

Wir lassen die LED drei Mal blinken.

```
1 #!/usr/bin/python3
2 from gpiozero import LED
3 from time import sleep
4 led1 = LED(23)
5 def blinke(n):
6     for i in range(n):
7         led1.on()  ; sleep(1)
8         led1.off() ; sleep(1)
9 blinke(3)
```

Es ist notwendig, den Rückgabewert der Funktion `LED()` zunächst in einer Instanz (`led1`) zu speichern. Diese Instanz hat die Methoden `on()` und `off()`, welche die Leuchtdiode ein und ausschalten. Die Funktion `sleep` hält die Programmausführung für die in Sekunden angegebene Zeitdauer an.

Für das Blinken einer Leuchtdiode gibt es die Methode `blink` mit vier Parametern. Die ersten beiden geben die Dauer eines Leuchtintervalls (Standard: 1) und die Dauer zwischen zwei Leuchtintervallen an (Standard: 1), jeweils in Sekunden (als Dezimalzahl). Der dritte Parameter gibt die Anzahl der Leuchtintervalle an (Standard: `None`); `None` bedeutet: ohne Ende. Der vierte Parameter erwartet einen Wahrheitswert (Standard: `True`), der festlegt, ob das Blinken im Hintergrund abläuft oder nicht. Bei `True` wird das Pythonprogramm schon während des Blinkens fortgesetzt. Bei `False` wird die Programmausführung erst nach der Beendigung des Blinkens fortgesetzt.[19]

[19] Wenn der dritte Parameter auf `None` gesetzt wurde, kann man darauf natürlich ewig warten.

```
1 #!/usr/bin/python3
2 from gpiozero import LED
3 led1=LED(23)
4 t_on=0.5 ; t_off=0.5
5 n=8 ; backgr=True
6 led1.blink(t_on,t_off,n,backgr)
7 print("Programm läuft weiter,")
8 input("ENTER beendet das Programm.")
```

Mithilfe der Klasse `gpiozero.PWMLED` kann man die LED dimmen (siehe Seite 130).

```
1 #!/usr/bin/python3
2 from gpiozero import PWMLED
3 from time import sleep
4 led1=PWMLED(23)
5 led1.value=0   ; sleep(3)   # dunkel
6 led1.value=0.5 ; sleep(3)   # halb hell
7 led1.value=1   ; sleep(3)   # ganz hell
```

Die Grundfrequenz ist beim Dimmen standardmäßig $f = 100\,\text{Hz}$.

Im folgenden Programm wird die Dateneingabe über einen Taster an GPIO 22 geprüft, siehe Abschnitt 1.2.4 auf Seite 35.

```
1 #!/usr/bin/python3
2 from gpiozero import Button
3 tst1 = Button(pin=22,pull_up=None,\
4 active_state=True,bounce_time=0.01)
5 while True:
6     if tst1.is_pressed: print("Schalter EIN")
7     else: print("Schalter AUS")
```

Bei der Definition einer Instanz vom Typ `Button` (hier: `tst1`) sollte man auf die genaue Setzung der Parameter achten. Im Beispiel werden externe Pullup- und Pulldown-Widerstände verwendet (siehe Abbildung 1.19 auf Seite 35). `active_state=True` bedeutet, dass das Programm einer Spannung von 3,3 V (HIGH) am GPIO den logischen Zustand True zuweist und 0 V (LOW) am GPIO den logischen Zustand False. Mit `active_state=False` wäre die Zuordnung umgekehrt. Die `bounce_time` dient dem Entprellen (hier: 10 ms).

Die Eigenschaft `is_pressed` ist `True`, während der Taster `tst1` gedrückt ist, sonst `False`. In einer Endlosschleife prüft das Programm immer wieder den Zustand des Tasters (Polling) und schreibt eine Meldung über den Zustand auf den Bildschirm..

Die Endlosschleife wird verlassen und das Programm endet, wenn der Nutzer `Ctrl-C` eingibt (siehe Seite 12).

Wir vereinen nun die in Abschnitt 1.2.4 auf Seite 35 beschriebene Schaltung zur Dateneingabe über einen Schalter an GPIO 22 und die in Abschnitt 1.2.4 auf Seite 32 beschriebene LED-Schaltung an GPIO 23 in einem Aufbau. Folgendes Programm lässt die LED genau dann leuchten, wenn der Schalter geschlossenen ist.

```
1  #!/usr/bin/python3
2  from gpiozero import Button,LED
3  from signal import pause
4
5  def ein1():
6      print("Schalter EIN") ; led1.on()
7  def aus1():
8      print("Schalter AUS") ; led1.off()
9
10 sch1 = Button(pin=22,pull_up=None,\
11 active_state=True,bounce_time=0.001)
12 led1=LED(pin=23,active_high=True,\
13 initial_value=False)
14
15 ein1() if sch1.is_pressed else aus1()
16 sch1.when_pressed  = ein1 # ohne ()
17 sch1.when_released = aus1 # ohne ()
18 pause()
```

Das Modul `signal` stellt die Funktion `pause()` bereit, mit der das Programm angehalten wird, bis der Nutzer `Ctrl-C` eingibt (siehe Seite 12). Im Beispiel werden externe Pullup- und Pulldown-Widerstände verwendet (siehe Abbildung 1.19 auf Seite 35). Die `bounce_time` zum Entprellen beträgt hier: 1 ms.

Die Zeile `ein1() if ...` bestimmt den Anfangszustand des Schalters mit einer einzeiligen `if-else`-Anweisung (siehe Seite 275). Die Eigenschaft `is_pressed` ist `True`, während `sch1` eingeschaltet ist, sonst `False`.

Die Methoden `when_pressed` und `when_released` reagieren auf eine Änderung des Zustands des Schalters. Die Überwachung des zugehörigen GPIO übernimmt das Betriebssystem, sodass im Programm keine Schleife mit fortlaufenden Zustandsabfragen (Polling) nötig ist. Dies Verfahren ist effizienter bezüglich der Belastung des Prozessors.

Man beachte, dass am Ende der beiden Zeilen mit den Methoden `when_pressed` und `when_released` der Funktionsname (`ein1` beziehungsweise `aus1`) ohne Klammern steht, da es ich um eine Referenz auf ein Funktionsobjekt und nicht um einen Aufruf handelt.

RPi.GPIO

GPIO können von einem Pythonprogramm über das Modul RPi.GPIO angesprochen werden, das mit dem Paket `python3-rpi.gpio` installiert wird. Es ist üblich, dem Modul beim Einbinden den Namen GPIO zu geben (`import RPi.GPIO as GPIO`). Die Version

des benutzen Moduls ist in der Konstanten `GPIO.VERSION` als Zeichenkette (`str`) gespeichert. Ein Dictionary (`dict`) namens `GPIO.RPI_INFO` enthält Information über den Raspberry Pi, zum Beispiel Modell, Prozessor und Arbeitsspeicher (RAM).

```
1 #!/usr/bin/python3
2 import RPi.GPIO as GPIO
3 print("GPIO-Version:",GPIO.VERSION)
4 print("Prozessor:",GPIO.RPI_INFO['PROCESSOR'])
```

Für die GPIO gibt es verschiedene Namensschemata: sie können über die Nummer des Pins auf der Stiftleiste J8 (beziehungsweise P1) angesprochen werden oder über die Nummer im BCM-Kanalnamen des System-on-a-Chip. Festgelegt wird dies mit der Methode `setmode`, welche die Konstante `GPIO.BOARD` (mit dem Wert 10) oder `GPIO.BCM` (mit dem Wert 11) als Argument bekommen kann. Wir verwenden die BCM-Kanalnamen: `GPIO.setmode(GPIO.BCM)`. Es ist bei den Methoden des Moduls üblich, die Namen der Konstanten (zum Beispiel `GPIO.BCM`) und nicht direkt deren Wert (zum Beispiel 11) zu übergeben, da dadurch der Programmcode für den Nutzer leichter lesbar ist.

Jeder GPIO kann entweder zur digitalen Eingabe (input) oder Ausgabe (output) benutzt werden. Die Methode `setup` legt für einen GPIO (erstes Argument der Methode) den Modus (zweites Argument der Methode) fest. Der Eingabe-Modus wird durch die Konstante `GPIO.IN` (Wert 1), der Ausgabe-Modus durch `GPIO.OUT` (Wert 0) festgelegt. Folgende Befehle bestimmen GPIO 22 zur Eingabe und GPIO 23 zur Ausgabe:

```
GPIO.setup(22,GPIO.IN)
GPIO.setup(23,GPIO.OUT)
```

Digitale Ausgabe bedeutet, dass eine logische 0 (0 V) oder 1 (3,3 V) an den GPIO gegeben wird. Der Ausgabewert kann durch eine Konstante `GPIO.LOW` (oder 0 oder `False`) oder `GPIO.HIGH` (oder 1 oder `True`) angeben werden. Folgendes setzt GPIO 23 auf logisch 1.

```
GPIO.output(23, GPIO.HIGH)
```

Am Ende eines Programms, das GPIO ansteuert, sollten die GPIO in den Standardzustand (Eingabe-Modus ohne Pullup- oder Pulldown-Widerstand) zurückgesetzt werden, in dem die irrtümliche Verbindung eines GPIO mit Masse (GND), 3,3 V oder einem anderen GPIO ungefährlich ist. Außerdem sollte dem Betriebssystem mitgeteilt werden, dass die GPIO nicht mehr benutzt (freigegeben) werden. Dies alles geschieht mit der Methode `cleanup`.

```
GPIO.cleanup()
```

Es ist aber meistens unschädlich, wenn der Aufruf von `cleanup` unterbleibt.

Wir gehen in folgendem Beispiel (Pythonprogramm `blink.py`) wieder von der in Abschnitt 1.2.4 auf Seite 32 beschriebenen Schaltung aus, in der eine Leuchtdiode (LED) über einen Vorwiderstand an GPIO 23 angeschlossen ist. Man kann auch die entsprechende Transistorschaltung nehmen, siehe Seite 34.

Wir starten das Programm (mit `python3 blink.py`), worauf gefragt wird, wie oft die LED blinken soll. Wir geben eine Zahl, zum Beispiel 10, ein. Dann erscheint der Text „Die LED sollte jetzt 10 mal blinken.“ und die LED sollte zehnmal aufleuchten.

```
#!/usr/bin/python3
# Programm blink.py zur Ansteuerung einer LED über GPIO
import RPi.GPIO as GPIO # importiere RPi.GPIO als GPIO
import time # importiere time (mit der Funktion sleep)
GPIO.setmode(GPIO.BCM) # GPIO-Namen (statt Pin-Nummern)
g = 23 # speichere die Nummer des GPIO-Ausgangs in g
GPIO.setup(g, GPIO.OUT) # setze GPIO g als Ausgang (OUT)
x = int(input("\nWie oft blinken? ")) # Eingabe von x
text1="\nDie LED sollte jetzt "+str(x)+"-mal blinken.\n"
print(text1) # (in Python 3 ist print eine Funktion)
n = 0 # speichere, wie oft die LED geblinkt hat, in n
try: # Folgende, solange nicht Ctrl-C gedrueckt wurde
   for i in range(0,x): # Schleife, x mal durchlaufen
      GPIO.output(g,GPIO.HIGH) # schalte den GPIO g ein
      n=i+1 # es hat wieder geblinkt
      time.sleep(0.2) # warte 0,2 Sekunden
      GPIO.output(g,GPIO.LOW) # schalte den GPIO g aus
      time.sleep(0.3) # warte 0,3 Sekunden
except KeyboardInterrupt: # Folgendes nach Ctrl-C
   GPIO.output(g,GPIO.LOW) # schalte den GPIO g aus
   text2="\nAbbruch nach "+str(n)+"-mal blinken.\n"
   print(text2) # (ab Python 3 ist print eine Funktion)
GPIO.cleanup() # GPIO wieder in den Standard-Zustand
```

Die Ausführung des Programms wird durch `Ctrl-C` abgebrochen (siehe Seite 12).

Der Programmteil nach `try:` wird ausgeführt, solange keine „Exception geworfen“ wird, also kein Fehler und kein Abbruchbefehl (allgemein: Ausnahme, englisch exception) auftritt. `Ctrl-C` bewirkt eine Exception, die ohne weitere Maßnahme zum sofortigen Programmabbruch führen würde. Hier jedoch wird eine geworfene Exception mit der Zeile `except KeyboardInterrupt: except` „gefangen“. Das Programm läuft dann mit dem Anweisungsblock nach der Zeile `except KeyboardInterrupt:` weiter. Ohne Exception wird dieser Anweisungsblock nicht durchlaufen.

Das folgende Programm benutzt RPI.GPIO um die LED zu dimmen (siehe Seite 130).

```
#!/usr/bin/python3
import RPi.GPIO as GPIO, time
def leuchte():
    global f
    po.ChangeFrequency(f)
    t="f = %3.0f Hz, Tastgrad = %4.2f"
```

```
7       for i in range(0,101,25):
8           po.ChangeDutyCycle(i)
9           print(t % (f,float(i)/100))
10          time.sleep(3)
11  GPIO.setmode(GPIO.BCM)
12  g = 23 ; GPIO.setup(g, GPIO.OUT)
13  po = GPIO.PWM(g, 1) ; po.start(0)
14  f = 10 ; leuchte()
15  f = 100 ; leuchte()
16  po.stop() ; GPIO.cleanup()
```

Der Befehl `po=GPIO.PWM(g, 1)` erzeugt eine Instanz namens `po` des PWM-Objekts zur Ausgabe auf dem GPIO namens `g` (hier: GPIO 23) mit der Grundfrequenz 1 Hz. Die Ausgabe wird mit dem Befehl `po.start(0)` begonnen, wobei der Tastgrad zunächst 0 ist (und die LED daher noch dunkel ist). Mit `f = 10` wird der gewünschte Wert der Grundfrequenz auf 10 Hz gesetzt.

In der Funktion `leuchte()` erfolgt eine Veränderung des PWM Signals. Der neue Frequenzwert wird mit dem Befehl `po.ChangeFrequency(f)` übergeben. Der Tastgrad wird in einer Schleife mit dem Befehl `po.ChangeDutyCycle(i)` stufenweise erhöht. Frequenz und Tastgrad werden auf dem Bildschirm formatiert ausgegeben und die Programmausführung wird dann mit `time.sleep(5)` für 5 s angehalten. Währenddessen leuchtet die LED mit dem eingestellten PWM Signal. Bei einer Frequenz von 10 Hz flackert die LED periodisch; diese Frequenz ist also zu niedrig für das Dimmen.

Mit `f = 100` wird die Grundfrequenz auf 100 Hz gesetzt und die Funktion `leuchte()` wird erneut aufgerufen. Man erkennt, dass bei dieser Frequenz kein periodisches Flackern auftritt. Dies liegt am Überschreiten der Verschmelzungsfrequenz für die visuelle Wahrnehmung.

Der Befehl `po.stop()` beendet die Ausgabe des PWM Signals.

Bei genauer Betrachtung fällt aber doch ein unregelmäßiges Flackern auch bei `f` = 100 Hz auf. Dies liegt an Unregelmäßigkeiten der vom Modul RPi.GPIO erzeugten, programmgesteuerten Modulation. Die Arbeit des Moduls wird nämlich unregelmäßig durch Aktivitäten des Betriebssystems unterbrochen, zum Beispiel für Garbage Collection . Es gibt Programme, welche PWM Signale unter Umgehung des Betriebssystems auf den GPIO ausgeben können, zum Beispiel *pi-blaster*. Damit ist dann auch ein Dimmen ohne unregelmäßiges Flackern möglich.

Im folgenden Programm wird die Dateneingabe über einen Taster an GPIO 22 geprüft, siehe Abschnitt 1.2.4 (Seite 35) und Abbildung 1.19.

```
1  # Programm "taster_gpio22" mit Python 3, AHG (2020)
2  import RPi.GPIO as GPIO # importiere RPi.GPIO als GPIO
3  from time import sleep # importiere sleep-Funktion
4  g1 = 22 # speichere die Nummer des GPIO-Eingangs in g1
5  GPIO.setmode(GPIO.BCM) # GPIO-Namen (statt Pin-Nummern)
```

```
GPIO.setup(g1, GPIO.IN) # setze GPIO g1 als Eingang (IN)
#
try: # Folgendes normalerweise, ohne Druecken von Ctrl-C
    while True: # Endlosschleife
        x = GPIO.input(g1) # weist x 0 oder 1 zu
        if x: # wenn x = 1, True / wenn x = 0, False
            print("Schalter EIN")
        else:
            print("Schalter AUS")
        sleep(0.5) # warte 0,5 Sekunden
except KeyboardInterrupt: # Folgendes nach Ctrl-C
    print("\nAbbruch durch Ctrl-C.\n")
GPIO.cleanup() # GPIOs zurueck in Standard-Zustand
```

Wir vereinen die in Abschnitt 1.2.4 auf Seite 35 beschriebene Schaltung zur Dateneingabe über einen Schalter an GPIO 22 und die in Abschnitt 1.2.4 auf Seite 32 beschriebene LED-Schaltung an GPIO 23 in einem Aufbau. Folgendes Programm lässt die LED bei geschlossenem Schalter fortlaufend das Notsignal SOS als Morsezeichen blinken. SOS-Anruf dreimal: ...|||... ...|||... ...|||... , Pause, dann wieder vorne beginnen.

```
#!/usr/bin/python
import RPi.GPIO as GPIO # importiere RPi.GPIO als GPIO
from time import sleep  # importiere sleep-Funktion
p=0.4 # kurze Morsesignallaenge (Dit) in Sekunden
g1 = 22  # Nummer des GPIO-Eingangs in Variable g1
g2 = 23  # Nummer des GPIO-Ausgangs in Variable g2
def kurz():
    GPIO.output(g2,GPIO.HIGH) # GPIO g2 ein
    sleep(p) # kurze Pause
    GPIO.output(g2,GPIO.LOW) # GPIO g2 aus
    sleep(p) # kurze Pause
def lang():
    GPIO.output(g2,GPIO.HIGH) # GPIO g2 ein
    sleep(3*p) # lange Pause
    GPIO.output(g2,GPIO.LOW) # GPIO g2 aus
    sleep(p) # kurze Pause
def sos():
    kurz() ; kurz() ; kurz()
    lang() ; lang() ; lang()
    kurz() ; kurz() ; kurz()
GPIO.setmode(GPIO.BCM)  # GPIO-Namen (statt Pin-Nummern)
GPIO.setup(g1, GPIO.IN)  # setze GPIO g1 als Eingang
GPIO.setup(g2, GPIO.OUT) # setze GPIO g2 als Ausgang
#
```

```
25 print("\nNotzeichen SOS blinken, Beenden mit Ctrl-C")
26 print("(SOS dreimal:  ...|||...  ...|||...  ...|||...)")
27 print("Uebertragungsrate %2.0f BpM" % (6.0/p))
28 try: # Folgendes normalerweise, ohne Ctrl-C
29     while True: # Endlosschleife
30         ein = GPIO.input(g1) # weist g1 eine 0 oder 1 zu
31         if ein: # wenn ein = 1 ist, True, sonst False
32             sos() # kurzkurzkurzlanglanglangkurzkurzkurz
33             sleep(2*p) # Pause auf 3 Dits verlaengern
34             sos() # kurzkurzkurzlanglanglangkurzkurzkurz
35             sleep(2*p) # Pause auf 3 Dits verlaengern
36             sos() # kurzkurzkurzlanglanglangkurzkurzkurz
37             sleep(6*p) # Pause auf 7 Dits verlaengern
38         else:
39             GPIO.output(g2,GPIO.LOW) # schalte g2 aus
40             sleep(0.01) # warte 0,01 s zum Entprellen
41 except KeyboardInterrupt: # Folgendes nach Ctrl-C
42     print("\nAbbruch durch Ctrl-C\n")
43 GPIO.cleanup() # GPIOs zurueck in den Standard-Zustand
```

8.2.10 Nutzung der Datenbusse

Über Datenbusse (1-Wire oder SPI) kann die Datenübertragung zwischen Raspberry Pi und externen Geräten verlaufen.

Temperatursensor DS18S20 am 1-Wire Bus

Der Eindraht-Datenbus 1-Wire wurde auf Seite 20 vorgestellt. Für Rechner mit einem 1-Wire Datenbus stellt der Linux Kernel das 1-wire (w1) subsystem zur Verfügung.

Zur Aktivierung des 1-Wire Datenbusses ruft man `raspi-config` auf (siehe Seite 42).

```
sudo raspi-config
```

und im Konfigurationsprogramm wählt man

```
(3) Interface Options
P7 1-Wire
```

Die Frage „Would you like the one-wire interface to be enabled?“ wird bejaht und nach <OK> und <Finish> sind wir schon fertig. Die Aktivierung wird erst nach einem Reboot (Neustart) des Raspberry Pi wirksam. Mit dem Befehl

```
lsmod | grep -i w1_
```

überprüft man, dass die kernel modules (Treiber) `w1_therm`, `w1_gpio` und `wire` geladen wurden. Die Option `-i` bewirkt, dass `grep` nicht zwischen Groß- und Kleinschreibung unterscheidet.

Wir benutzen hier den Temperatursensor DS1820 (siehe Seite 30) und gehen davon aus, dass die in Abbildung 1.20 (Seite 36) gezeigte Schaltung aufgebaut wurde. Statt des Temperatursensors DS18S20 kann auch der neuere Typ DS18B20 verwendet werden, welcher die Temperatur mit höherer Auflösung (12 bit) ausgibt.

`sysfs` ist ein virtuelles Dateisystem des Kernels, welches im Verzeichnis `/sys/bus/` Datenbusse abbildet. Mit

```
cd /sys/bus/w1/devices/
```

wechseln wir in ein Verzeichnis für 1-Wire Geräte und lassen uns mit dem Befehl

```
ls
```

den Inhalt des Verzeichnisses anzeigen. Wir wechseln in das Unterverzeichnis, dessen Name `SERIENNUMMER` der Seriennummer des Sensors entspricht

```
cd SERIENNUMMER
```

und geben den Inhalt der Datei `temperature` mit dem Befehl `cat` auf dem Monitor aus.

```
cat temperature
```

Es wird eine ganze Zahl ausgegeben. Fügt man nach den beiden ersten Ziffern in Gedanken ein Dezimalkomma ein, erhält man die Temperatur in °C.

Wenn man den Sensor mit zwei Fingern anfasst, erwärmt er sich und ein erneuter Befehl `cat temperature` sollte eine höhere Temperatur anzeigen.

Die Datei `/sys/bus/w1/devices/SERIENNUMMER/temperature` wird in folgendem Pythonprogramm fortlaufend ausgelesen und die Temperatur in °C auf dem Bildschirm ausgegeben, bis zum Abbruch durch `Ctrl-C` (siehe Seite 12). Die Seriennummer im Dateinamen, die mit `10-` beginnt, wird mit der Funktion `glob.glob()` automatisch ergänzt.

```
1  # temp_onewire.py : temperature measurement with DS18S20
2  from glob import glob # um den Pfad zum Sensor zu finden
3  import time   # kein Modul für den 1-Wire Bus notwendig
4  s = glob("/sys/bus/w1/devices/10-*")[0]+"/temperature"
5  print("\nTemperaturmessung, Abbruch durch Ctrl-C")
6  print("****************************************")
7  try: # Folgendes normalerweise, ohne Drücken von Ctrl-C
8      while True: # Endlosschleife zur Temperaturmessung
9          zeit = time.strftime("%H:%M:%S") # Uhrzeit
10         with open(s,"r") as f: T_roh = f.read() # T_roh
11         T = round(float(T_roh)/1000,1) # Temperatur T
12         print(zeit+": T = %4.1f 'C" % T) # Ausgabe
13         time.sleep(3) # warte 3 s bis nächste Messung
14 except KeyboardInterrupt: # nach dem Drücken von Ctrl-C
15     print("\n- Messung beendet -\n") # Programm-Ende
```

Obiges Programm kann erweitert werden, wie im nachfolgenden Beispiel.

```
# temp_onewire_plot.py : temperature curve with DS18S20
from glob import glob # um den Pfad zum Sensorzu finden
from subprocess import run # Zugriff auf Betriebssystem
import time   # kein Modul für den 1-Wire Bus notwendig
s = glob("/sys/bus/w1/devices/10-*")[0]+"/temperature"
v = "T-"+time.strftime("%y%m%d%H%M%S") # Namensanfang
dat_name = v + ".dat"   ;   plt_name = v + ".png"
td = open(dat_name,'w') # Dateiobjekt für Temp.daten

print("\nTemperaturmessung, Abbruch durch Ctrl-C")
print("in der Graphik-Datei",plt_name)
print("Beenden der Graphik-Anzeige fbi durch q")
print("***************************************")
iv = int(input("Zeit in s zwischen zwei Messungen: "))

try: # Folgendes normalerweise, ohne Drücken von Ctrl-C
    while True: # Endlosschleife zur Temperaturmessung
        zeit = time.strftime("%d.%m.%y/%H:%M:%S") # Zeit
        with open(s,"r") as f: T_roh = f.read() # T_roh
        T = round(float(T_roh)/1000,2) # Temperatur T
        temp = "%5.1f 'C" % round(T,1) # T, gerundet
        print(temp,end="\r",flush=True) # Ausgabe von T
        td.write(zeit+' '+ "%6.2f" % round(T,2) +'\n')
        time.sleep(iv) # Warten in s bis nächste Messung
except KeyboardInterrupt: # nach dem Drücken von Ctrl-C
        td.close() # Schließen der Temperaturdaten-Datei

gd = open("batch",'w') # Dateiobjekt für Gnuplot
aus = "set encoding iso_8859_1\n"
aus = aus + "set title 'DS18S20, Messung " + v + \
", Mess-Intervall " + str(iv) +" s'\n"
aus = aus + 'set format x "%H:%M\\n %S s"\n'
aus = aus + "set ylabel 'Temperatur [\260C]'\n"
aus = aus + "set xdata time\n"
aus = aus + "set timefmt '%d.%m.%y/%H:%M:%S'\n"
aus = aus + "set terminal pngcairo mono\n"
aus = aus + "set output '" + plt_name + "'\n"
aus = aus + "p '" + dat_name + "' using 1:2 w l t ''"
gd.write(aus) ; gd.close() # Schreiben, Schließen und
run(["gnuplot","batch"])   # Ausführen der Gnuplot-Datei
run(["fbi",plt_name])      # Anzeige der Graphik-Datei
```

In diesem Programm wird das Zeitintervall zwischen zwei Messungen (`iv`) vom Nutzer vorgegeben und die Messwerte werden in einer Datei gespeichert. Nach Abbruch der Messung (mit `Ctrl-C`) wird mit Gnuplot (siehe Seite 250) eine Graphik erstellt, gespeichert und mit `fbi` (siehe Seite 173) angezeigt (Beenden mit `q`).

A/D-Wandler MCP3208 am SPI Bus

Der Datenbus SPI wurde auf Seite 21 vorgestellt.

Für Rechner mit einem SPI Datenbus stellt der Linux Kernel eine API (Schnittstelle zur Programmierung von Anwendungen) bereit, über die Programme auf den SPI Bus zugreifen können. Um diese mit einem Pythonprogramm leichter nutzen zu können, gibt es eine Anpassung (python binding) in Form des Moduls `spidev`.

Zur Aktivierung des SPI Busses ruft man `raspi-config` auf (siehe Seite 42).

```
sudo raspi-config
```

und im Konfigurationsprogramm wählt man

```
(3) Interface Options
P4 SPI
```

Die Frage „Would you like the SPI interface to be enabled?“ wird bejaht und nach <OK> und <Finish> sind wir schon fertig.

Nach einem Neustart des Betriebssystems

```
sudo reboot
```

werden die Treibermodule (`spidev` und `spi_bcm2835` oder ähnlich) über den Gerätebaum zur Verfügung gestellt. Man kann dies mit dem Befehl

```
lsmod | grep spi
```

prüfen.[20] Es sollten zwei Zeilen ausgegeben werden, die mit `spi_bcm` beginnen.

Wir benutzen hier den A/D-Wandler MCP3208 (siehe Seite 31) und gehen davon aus, dass die in Abbildungen 1.22 (Seite 38) und 8.11 gezeigte Schaltung aufgebaut wurde.

Folgendes Programm misst die Spannung am Eingang CH2 des MCP3208 und gibt das Ergebnis einmal pro Sekunde auf dem Bildschim aus. Das Programm kann ohne Root-Rechte ausgeführt werden. Drücken von Ctrl-C beendet das Programm (siehe Seite 12).

```
1 # mcptest.py : test measurement of voltage with MCP3208
2 import spidev
3 from time import sleep
4 #
5 print("\n--- Program mcp3208easy.py ---")
```

[20] Der Befehl `lsmod` gibt eine Liste aller geladenen Module aus. Der Ausgabedatenstrom wird mithilfe des |-Zeichens (das eine Pipe erzeugt; es ist meistens auf der Tastatur vorhanden) als Eingabe für den Befehl `grep spi` verwendet, der alle Zeilen ausgibt, welche die Zeichenkette spi enthalten.

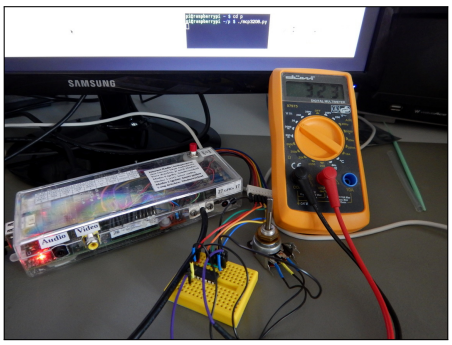


Abbildung 8.11: Spannungsmessung mit Raspberry Pi 1 B und A/D-Wandler MCP3208

```
6  print("Input on channel CH2 of mcp3208: U max. 3.3V\n")
7  #
8  # Variablendeklaration und Oeffung des SPI-Busses
9  #
10 # Auswahl des SPI_Busses, hier nur 0 moeglich
11 SPI_Bus = 0
12 # Slave, der mit CS-Signal angesprochen wird
13 CS = 0
14 # Frequenz in Hz fuer das SCLK-Signal (clock)
15 f = 20000
16 # Referenzspannung in V an Pin 15 des MCP3208
17 Uref = 3.3
18 # Spannungsintervall digitaler Wert-Einheiten
19 Uiv = Uref / 4096
20 # Kanalnummer (Bereich 0 - 7) des AD-Wandlers
21 K = 2 # Kanal CH2 des AD-Wandlers
22 # Pausendauer zwischen Messsungen in Sekunden
23 Pausendauer = 1
24 # Erzeugung eines SpiDev-Objekts namens spi
25 spi = spidev.SpiDev()
26 # Oeffnung von SPI-Bus SPI_Bus fuer Slave CS
27 spi.open(SPI_Bus, CS)
28 # Vorgabe der maximalen SPI-Bus-Taktfrequenz
29 spi.max_speed_hz = f
30 #
31 # Ausgabe eines Textes vor Beginn der Messung
32 print("Measurement of voltage on CH2 of MCP3208")
33 print("(change with potentiometer, Ctrl-C to quit)\n")
34 #
35 # Folgendes normalerweise, ohne Druecken von Ctrl-C
36 try:
```

```
37 #
38 # Endlos-Schleife zur Messung und Ausgabe des Wertes
39 #
40     while True:
41         adc = spi.xfer2([6 + (K >> 2), K << 6, 0])
42         # je 3 Bytes senden (MOSI) und empfangen (MISO)
43         Wert = ((adc[1]&15) << 8) + adc[2] # Bits->Zahl
44         U = Uiv * Wert + Uiv/2  # Digitalwert->Spannung
45         print("K {0:1d}: U = {1:4.2f} V".format(K,U))
46         sleep(Pausendauer)  # Pause zwischen Messungen
47 #
48 # Folgendes nach Druecken von Ctrl-C
49 except KeyboardInterrupt:
50     spi.close()  # trennt SpiDev-Objekt spi vom SPI-Bus
51     print("\n- Messung beendet -\n")  # Programmende
```

Schauen wir uns den Programmcode der Reihe nach an. Nach Einbindung der notwendigen Module (`spidev`, `time`) werden zunächst einige Werte in Variablen gespeichert, auf die wir später zurückgreifen.

Grundsätzlich wäre es möglich, mehrere SPI Busse am Raspberry Pi zu betreiben, die (beginnend mit 0) durchnummeriert werden. Hier gibt es nur den SPI Bus mit der Nummer 0, dessen Nutzung in der Zeile `SPI_Bus = 0` festgelegt wird.

Kommunikation auf einem SPI Datenbus ist asymmetrisch gemäß der Master/Slave Technologie. In der Elektronik werden die Bezeichnungen dieser Technologie ohne die sozialgeschichtlich bedenklichen Assoziationen verwendet. Es gibt jedoch Bestrebungen zu einer besseren Namensgebung.

Der SPI Bus 0 könnte verschiedene Slaves über jeweils eine `CS` Datenleitung ansprechen. Hier wird nur der Slave MCP3208 über die erste `CS` Datenleitung mit der Nummer 0 angesprochen, was in der Zeile `CS = 0` festgelegt wird.

Der SPI Bus kann mit unterschiedlichen Taktfrequenzen betrieben werden und die meisten der angeschlossenen Geräte (Slaves) können mit unterschiedlichen, auch wechselnden Taktfrequenzen (und sogar Änderungen im Tastgrad) arbeiten.

Bei dem hier betrachteten AD-Wandler wird der Frequenzbereich durch die Dauer der Wandlung beschränkt, die 12 Takte umfasst. Die höchste mögliche Frequenz liegt unter optimalen Bedingungen bei 2 MHz, so dass maximal 100 000 Messwerte pro Sekunde erfasst werden können, wenn der Master dies auch schafft.

Es gibt auch eine Höchstdauer für die sichere Zwischenspeicherung des analogen Spannungswerts in einem internen Kondensator während der Wandlung von etwa 1,2 ms unter ungünstigen Bedingungen. Daraus ergibt sich eine Frequenz von mindestens 10 kHz.

Da in obigem Programm keine besonders schnelle Datenerfassung notwendig ist, wird die gewünschte maximale Taktfrequenz in Hz mit `f = 20000` vorgegeben. Man kann die Taktfrequenz auf der `SCLK` Signalleitung mit Hilfe eines Oszilloskops sichtbar machen.

Der Spannungsbereich, in dem vom MCP3208 gemessen und gewandelt wird, liegt zwischen 0 V und VREF. Die Referenzspannung VREF kann höchstens so groß wie die

Versorgungsspannung VDD sein und beträgt in obiger Schaltung 3,3 V. Dies wird mit der Zeile `Uref = 3.3` angegeben.

Der MCP3208 ist ein 12-Bit Wandler. Der Messspannungsbereich wird in $2^{12} = 4096$ gleich lange Einzelintervalle zerlegt, die von 0 bis 4095 durchnummeriert werden. Die Länge eines Einzelintervalls beträgt `VREF` / 4096. Dieser Wert wird mit der Zeile `Uiv = Uref / 4096` in der Variablen `Uiv` gespeichert.

Die acht Eingangskanäle sind von 0 bis 7 nummeriert. Mit der Zeile `Kanal = 2` wird der Eingangskanal 2 namens `CH2` festgelegt.

Da wir eine Messung pro Sekunde durchführen wollen, wird mit der Zeile `Pausendauer = 1` ein Wert von 1 für die Dauer zwischen Messungen in Sekunden festgelegt.

Das Modul `spidev` enthält eine Klasse `SpiDev`. Die Zeile `spi = spidev.SpiDev()` erzeugt eine Instanz namens `spi`, mit der man einen SPI Bus ansteuern kann.

In der Zeile `spi.open(SPI_Bus, CS)` wird mit der Methode `open` für die Instanz `spi` ein bestimmter SPI Bus geöffnet und der anzusprechende Slave angegeben.

In der Zeile `spi.max_speed_hz = f` wird die gewünschte maximale Taktfrequenz festgelegt. Die tatsächliche Taktfrequenz liegt nicht über dem angegebenen Wert, kann aber darunter liegen, weil a) der SPI Bus die Frequenz nur schrittweise ändern kann und b) der Prozessor zu langsam für die gewünschte maximale Taktfrequenz sein könnte.

Die Zeilen `try:` und `except KeyboardInterrupt:` ermöglichen, das Programm mit `Ctrl-C` abzubrechen und dann „Aufräumarbeiten“ durchführen zu können.

Messung und Anzeige der Analogspannung geschehen im Anweisungsblock einer Endlosschleife, die mit der Zeile `while True:` eingeleitet wird.

In der Zeile `adc = spi.xfer2([6 + (K » 2), K « 6, 0])` wird die Methode `xfer2` des `spi` Objekts mit einem Argument aufgerufen, das aus einer dreiteiligen Liste besteht, und der Rückgabewert, der ebenfalls aus einer dreiteiligen Liste besteht, wird in der Variablen `adc` gespeichert.

Die Methode `xfer2` sorgt für die Datenübertragung zwischen Master und Slave. Sie kann nur ganzzahlige Vielfache eines Bytes vom Master zum Slave (MOSI) senden und liest gleichzeitig dieselbe Anzahl von Bytes von der anderen Datenleitung (MISO).

Da für einen Messwert mindestens 19 zusammenhängende Bitpaare (Paar bedeutet: 1 Bit auf MOSI und 1 Bit auf MISO) übertragen werden müssen (siehe unten), sind drei Byte-Paare (24 Bitpaare) notwendig, denn zwei Byte-Paare (16 Bitpaare) wären zu wenig. Daher enthalten beide Listen der Methode `xfer` (Argument und Rückgabewert) jeweils drei Byte.

Die 19 Bitpaare für eine Messwertabfrage bestehen aus einem Befehl vom Master an den Slave (5 Bits auf der `MOSI` / $\mathtt{D_{IN}}$ Leitung), einer Pause zum Einlesen eines Messwerts (2 Bits) und dem Datenwort, das den Messwert darstellt, welches vom Slave zum Master gesendet wird (12 Bits auf der $\mathtt{D_{OUT}}$ / `MISO` Leitung).

Die 5 Bits des Befehls bestehen aus einem Startbit, `S` (1 = high), einem Bit zur Festlegung der Messart, `U` (1 = high für unipolar oder 0 = low für bipolar, hier: 1) und drei Bits zur Angabe des Messkanals, `K K K` (hier: binär `010` = dezimal 2).

Zur Auffüllung auf 3 Byte-Paare werden bei der Messwertabfrage den 19 Bits auf der `MOSI` Leitung fünf Bits vorangestellt, die nur low = 0 enthalten. Die Abfolge der Bits auf den `MOSI` und `MISO` Leitungen wird in Abbildung 8.12 veranschaulicht.

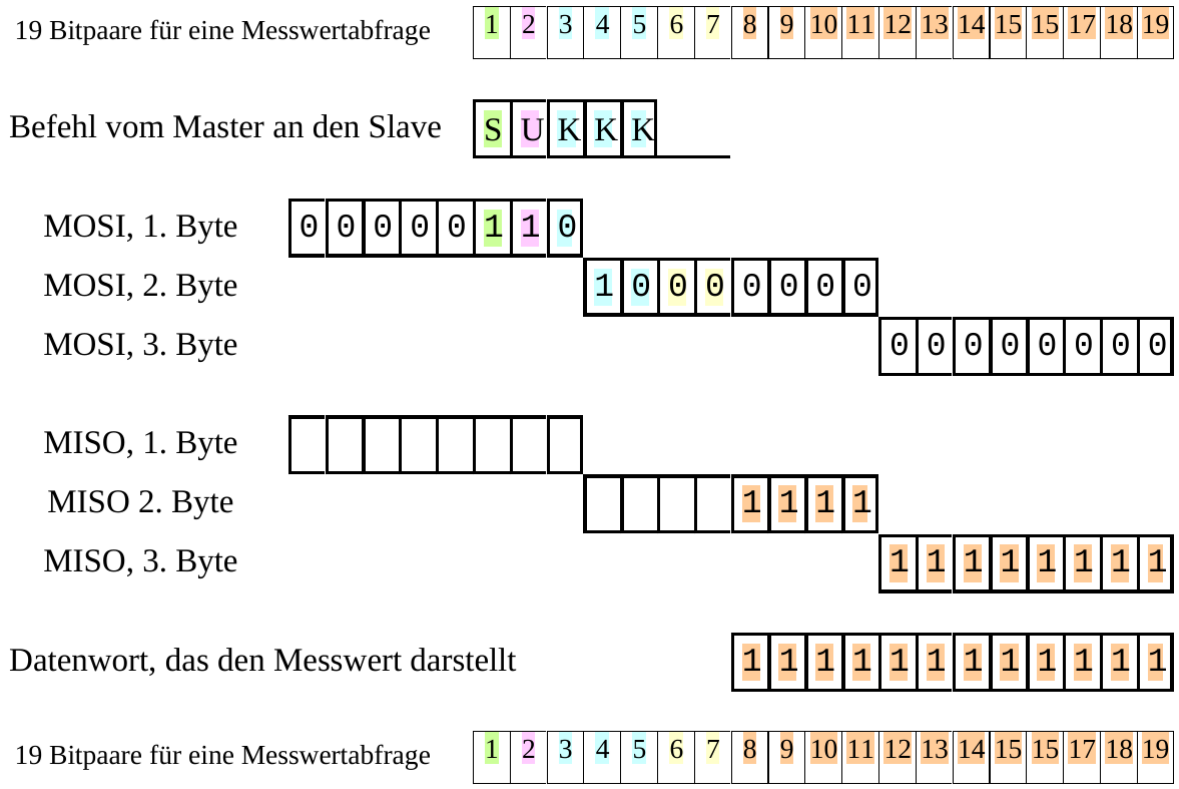


Abbildung 8.12: Datenübertragung auf dem SPI Datenbus mit der Methode `xfer2`

Die ersten 12 Bits auf der `MISO` Leitung (welche dem 12 Bit-Datenwort vorausgehen) sind unwichtig (werden nicht ausgewertet) und wurden daher in Abbildung 8.12 nicht angegeben. Das Datenwort besteht in diesem Beispiel aus der Bitfolge `111111111111`, was dezimal der Zahl 4095 entspricht.

Die Methode `xfer2` erwartet als Argument eine Liste von drei Bytes.

Das erste Byte (8 Bits) bekommt an der 8. Stelle (LSB) das höchstwertige Bit der Kanalnummer. Dies geschieht, indem von den drei Bits, welche die Kanalnummer (dezimal zwischen 0 und 7) repräsentieren (hier: `010`), mit der Operation `K >> 2` die beiden letzten Bits entfernt werden und das erste Bit (hier: 0) an die niederwertigste Stelle rückt. Dann werden durch Addition von `6 = 4 + 2` die 6. Stelle (dezimal 4) und die 7. Stelle (dezimal 2) jeweis mit einer 1 besetzt, so dass hier das Byte `00000110` entsteht.

Das zweite Byte bekommt an der 1. Stelle (MSB) und der 2. Stelle das zweite (hier: 1) beziehungsweise dritte Bit der Kanalnummer (hier: 0). Dies geschieht, indem von den drei Bits, welche die Kanalnummer repräsentieren (hier: `010`), mit der Operation `K << 6` das erste Bit entfernt wird, das zweite Bit um sechs Stellen nach links an die 1. Stelle des Bytes, und das dritte Bit der Kanalnummer um sechs Stellen nach links an die 2. Stelle des Bytes gerückt werden. So wird hier das Byte `10000000` gebildet.

Das dritte Byte wird Null gesetzt, dies entspricht `00000000`.

Mit dem oben beschriebenen (durch drei Bytes repräsentierten) Befehl gibt die Methode `xfer2` als Rückgabewert eine Liste von drei Bytes (namens `adc`), die den Messwert enthält.

In der Zeile `Wert = ((adc[1]&15) << 8) + adc[2]` wird aus der Liste `adc` die Variable `Wert` gebildet, welche das Einzelintervall des Messspannungsbereichs des 12-Bit Wandlers bezeichnet (siehe oben). Es gilt daher $0 \leq \text{Wert} \leq 4095$.

Das erste Byte der zurückgegebenen Liste, `adc[0]`, wird verworfen.

Vom zweiten Byte, `adc[1]`, werden, mithilfe der Operation `adc[1]&15`, die 4 niederwertigsten Bits genommen (dezimal 15 entspricht binär `00001111` und der Operator `&` erzeugt eine bitweise UND-Verknüpfung). Diese werden, mithilfe der Operation `<< 8`, um 8 Stellen nach links verschoben, so dass sie die ersten 4 Stellen einer zwölfstelligen Binärzahl darstellen.

Zu den ersten 4 Stellen der zwölfstelligen Binärzahl wird das dritte Byte, `adc[2]`, addiert (letzte 8 Stellen der zwölfstelligen Binärzahl), so dass die zwölfstellige Binärzahl fertig ist und ihr Wert der Variablen namens `Wert` zugewiesen werden kann.

In der Zeile `U = Uiv * Wert + Uiv/2` wird aus der Variablen `Wert` ein analoger Spannungswert (Messwert in V) berechnet.

Der zu einem Einzelintervall `Wert` gehörige analoge Spannungswert liegt zwischen `Wert · (VREF / 4096)` und `(Wert + 1) · (VREF / 4096)` und wir ordnen dem Einzelintervall `Wert` daher den Mittelwert `(Wert + Wert + 1) · (VREF / 4096) / 2 = Wert · (VREF / 4096) + (VREF / 4096) / 2` zu. Dies geschieht in der Zeile $U = Uiv * Wert + Uiv/2$ und der so gefundene analoge Spannungswert (in V) wird in der Variablen `U` gespeichert.

Mit `print("K 0:1d: U = 1:4.2f V".format(K,U))` werden die Werte der Variablen `K` (einstellige Ganzzahl, `d`) und `U` (vierstellige Gleitkommazahl, `f`, mit zwei Nach-

kommastellen) formatiert als Teil einer Zeichenkette auf dem Bildschirm ausgegeben. Formatnummer `0` gibt eine Ganzzahl (`d`) einstellig aus; Formatnummer `1` gibt eine Gleitkommazahl (rationale Zahl, `f`) mit vier Zeichen aus, wovon das erste Zeichen die Vorkommazahl, das zweite den Dezimalpunkt und das dritte und vierte Zeichen zwei Nachkommastellen zeigen.

Die Zeile `sleep(Pausendauer)` ruft die Methode `sleep` der Klasse `time` auf, um (zwischen zwei Messungen) die Programmausführung für `Pausendauer` Sekunden anzuhalten.

Nach Drücken von `Ctrl-C` wird mit der Zeile `spi.close()` der SPI Bus freigegeben, der bisher von der Instanz `spi` belegt war.

In der letzten Zeile `print("\n- Messung beendet -\n")` wird eine Meldung an den Nutzer ausgegeben.

Aktivierung des I²C Busses

Über den I²C Datenbus (siehe Seite 21) können verschiedene Geräte an den Raspberry Pi angeschlossen werden. Um ihn nutzen zu können, müssen entsprechende Module im Gerätebaum (device tree) zur Verfügung stehen. Dies erreicht man einfach, indem man `raspi-config` aufruft (siehe Seite 42)

```
sudo raspi-config
```

und im Konfigurationsprogramm wählt man

```
(3) Interface Options
I5 I2C
```

Die Frage „Would you like the ARM I2C interface to be enabled?“ wird bejaht und nach <OK> und <Finish> sind wir schon fertig. Nach einem Neustart des Betriebssystems

```
sudo reboot
```

können wir auf den I²C Datenbus zugreifen. Der Befehl

```
lsmod | grep i2c
```

sollte zwei Zeilen ausgeben, mit `i2c_bcm2835` beziehungsweise `i2c_dev` am Anfang.

Der Befehl `lsmod` gibt eine Liste aller geladenen Module aus. Der Ausgabedatenstrom wird mit dem |-Zeichens (das eine Pipe erzeugt; es ist meistens auf der Tastatur vorhanden) als Eingabe des Befehls `grep i2c` verwendet, der alle Zeilen ausgibt, welche die Zeichenkette `i2c` enthalten. Im Ergebnis sollten zwei Zeilen am Bildschirm erscheinen, die mit `i2c` beginnen.

Die I²C Kanäle bekommen wir mit dem Befehl

```
ls -l /dev/i2c*
```

genannt. Hier sollte `/dev/i2c-1` für Kanal 1 erscheinen.

Wir installieren nun ein Programm, das den I²C Bus mit Befehlen in der Kommandozeile anspricht (`i2c-tools`) und eine Python-Biblothek zur Ansteuerung des I²C Busses (`python3-smbus`).

```
sudo apt install i2c-tools python3-smbus
```

Um den I²C Bus als normaler Nutzer ohne Administratorrechte verwenden zu können, müssen wir zur Gruppe `i2c` gehören. Das sollte bereits der Fall sein. Wir prüfen dies mit dem Befehl

```
groups
```

welcher die Gruppen anzeigt, denen wir angehören. Wenn wir noch nicht Mitglied der Gruppe `i2c` sind, treten wir mit dem Befehl

```
sudo adduser $USER i2c
```

der Gruppe `i2c` bei. Damit obige Änderungen wirksam werden, starten wir den Raspberry Pi neu. Mit dem Befehl

```
i2cdetect -y 1
```

sehen wir, welche Geräte am ersten I²C Bus angeschlossen sind. Die Option `-y` dient zur Unterdrückung eines Warnhinweises und einer Rückfrage.

Temperatursensor LM75 am I²C Bus

Der Temperatursensor LM75 wurde bereits vorgestellt (siehe Seite 30) und ebenso der I²C Datenbus (siehe Seite 21). Die elektronische Schaltung zum Anschluss des LM75 an den I²C Bus des Raspberry Pi wurde ab Seite 36 beschrieben. Wenn Pins 7, 6 und 5 des LM75 mit Masse verbunden sind, ist die Adresse des LM75 auf dem I²C Bus `0x48`.

Um zu sehen, ob der LM75 am ersten I²C Bus angeschlossen ist, geben wir den Befehl

```
i2cdetect -y 1
```

Wenn der LM75 mit der Standardadresse `0x48` angeschlossen ist, sollte in der mehrzeiligen Antwort des Programms eine 48 in einer mit 40 beginnenden Zeile enthalten sein. Die Adresse wird als Hexadezimalzahl angegeben, erkennbar am Präfix `0x`. Hexadezimal 48 entspricht dezimal $4 \cdot 16 + 8 = 72$.

Der I²C Datenbus kann direkt über den Befehl `i2cget` angesprochen werden oder in einem Pythonprogramm mithilfe der Bibliothek `python3-smbus` zur Ansteuerung des I²C Busses. Im ersten Beispiel benutzen wir den den Befehl `i2cget`, um vom Temperatursensor LM75 ein Datenwort (2 Byte) auszulesen, welches die Temperatur angibt.

Eine Abfrage von Rohdaten (codierte Daten, wie sie vom LM75 geliefert werden) geschieht mit dem Befehl

```
i2cget -y 1 0x48 0x00 w
```

und wir erhalten eine Antwort, welche in codierter Form den Temperaturwert angibt, zum Beispiel `0xf718`. Diese Antwort besteht aus einer vierstelligen hexadezimalen Zahl (Präfix `0x`). Die beiden letzten Stellen, hier `18`, ergeben dezimal den ganzzahligen Anteil (ohne Nachkommastelle) des Temperaturwerts, hier $1 \cdot 16 + 8 = 24$. Für die aktuelle Temperatur T gilt also $24\,°C \leq T < 25\,°C$. Eine vollständige Auswertung der Rohdaten vom LM75 wollen wir mit einem Python-Programm vornehmen.

Wir schreiben mit einem Editor folgendes Pythonprogramm und speichern es unter dem Namen `lm75_1.py` im Verzeichnis `/home/ahg/progs`

```
1  #!/usr/bin/env python3
2  from subprocess import Popen, PIPE
3  from time import sleep
4  #
5  def T(wert):
6      grad = wert & 0xFF
7      gradnachkomma = wert >> 15
8      if (grad & 0x80) == 0x80:
9          grad = -((~grad & 0xFF) + 1)
10     temp = grad + gradnachkomma * 0.5
11     print(str(temp)+' Grad C')
12 #
13 print("\nTemperatur am LM75, Abbruch durch Ctrl-C")
14 print("****************************************")
15 L = [ 'i2cget' , '-y' , '1' , '0x48' , '0x00' , 'w' ]
16 #
17 try:
18     while True:
19         p = Popen(L , stdout=PIPE)
20         T( int(p.stdout.read(), 16) )
21         sleep(1)
22 except KeyboardInterrupt:
23     print ("\n- Messung beendet -\n")
```

Die erste Zeile ist eine Shebang-Zeile (siehe Seite 261). Funktion `T` wandelt die Rohdaten in einen Temperaturwert und gibt ihn aus. Im Einzelnen geschieht folgendes in der Funktion: In Zeile 6 wird der Referenz `grad` das untere Byte des Rohdatenwerts zugewiesen und in Zeile 7 wird der Referenz `gradnachkomma` das obere Byte des Rohdatenwerts zugewiesen. In Zeile 8 wird geprüft, ob das höchstwertige Bit von `grad` 1 ist. Ist das der Fall, ist der Temperaturwert negativ und Zeile 9 bildet das negative Zweierkomplement von `grad`. In Zeile 10 werden Vor- und Nachkommatemperaturzahl addiert und die Temperatur in °C mit der Referenz `temp` gespeichert. Zeile 11 schreibt die Temperatur mit einer Auflösung von 0,5 °C auf den Bildschirm.

In den Zeilen 13 und 14 geschieht eine einfache Bildschirmausgabe. In Zeile 15 wird der Shell-Befehl `i2cget` vorbereitet, indem seine Bestandteile in einer Liste `L` gespeichert

werden. Dieser Shell-Befehl liest die Rohdaten vom LM75 über den I²C Bus ein. Die Option `-y` dient zur Unterdrückung eines Warnhinweises und einer Rückfrage, die 1 steht für die Kanalnummer des I²C Busses. Die Option `0x48` ist die Adresse des LM75 auf dem Bus (sie liegt allgemein zwischen `0x003` und `0x77`). Die `0x00` bezeichnet das erste interne Register des LM75 (Registeradressen liegen allgemein zwischen `0x00` und `0xFF`). Der LM75 hat vier Register. Das erste ist ein 16-Bit-Register und enthält die Temperatur als Rohdaten. Das `w` am Ende steht für den Modus „word“, das heißt, es werden 2 Byte = 16 Bit gelesen (im Modus `b` würde nur 1 Byte = 8 Bit gelesen).

In Zeilen 17 und 23 wird eine Ausnahmebehandlung erzeugt, um auf das Drücken von `Ctrl-C`, das ist die Ausnahme `KeyboardInterrupt`, zu reagieren.

In Zeilen 18 bis 21 läuft eine Endlosschleife, bis es ein `KeyboardInterrupt` gibt. In Zeile 19 führt die Methode `Popen` den Shell-Befehl `i2cget` aus und speichert die Antwort (String). In Zeile 20 wird die Funktion `T` aufgerufen, wobei die Rohdaten als Ganzzahl das übergebene Argument sind. In Zeile 21 wird eine Sekunde gewartet. Dies ist das Zeitintervall zwischen zwei Temperatur-Ausgaben.

Zeile 23 wird nach dem Drücken von `Ctrl-c` ausgeführt und macht eine einfache Bildschirmausgabe. Danach endet das Programm.

Das Programm wird mit dem Befehl

```
python3 /home/ahg/progs/lm75_1.py
```

ausgeführt.

Obiges Programm benutzt den Befehl `i2cget`, um über den I²C Bus vom Temperatursensor ein Datenwort (2 Byte) auszulesen. Im folgenden Programm wird dafür die Bibliothek `smbus` verwendet.

```
1  #!/usr/bin/env python3
2  import smbus
3  from time import sleep
4  #
5  def T(wert):
6      grad = wert & 0xFF
7      gradnachkomma = wert >> 15
8      if (grad & 0x80) == 0x80:
9          grad = -((~grad & 0xFF) + 1)
10     temp = grad + gradnachkomma * 0.5
11     print(str(temp)+' Grad C')
12 #
13 i2cbus=smbus.SMBus(1)
14 #
15 print("\nTemperatur am LM75, Abbruch durch Ctrl-C")
16 print("****************************************")
17 #
18 try:
```

```
19      while True:
20          daten = i2cbus.read_word_data(0x48, 0x00)
21          T(daten)
22          sleep(1)
23  except KeyboardInterrupt:
24      print ("\n- Messung beendet -\n")
```

In Zeilen 5 bis 11 wird die Funktion `T` definiert, welche die Rohdaten in einen Temperaturwert wandelt und ihn ausgibt. Sie ist identisch mit der gleichnamigen Funktion aus dem Programm `lm75_1.py`, siehe oben.

In Zeile 13 wird eine Instanz für den I²C Bus 1 gebildet und der Referenz `i2cbus` zugewiesen. In den Zeilen 15 und 16 geschieht eine einfache Bildschirmausgabe.

In Zeilen 18 und 23 wird eine Ausnahmebehandlung erzeugt, um auf das Drücken von `Ctrl-C`, das ist die Ausnahme `KeyboardInterrupt`, zu reagieren.

In Zeilen 19 bis 22 läuft eine Endlosschleife, bis es ein `KeyboardInterrupt` gibt. In Zeile 20 holt die Methode `read_word_data` ein Wort (2 Byte) vom Sensor, welches die Rohdaten als Ganzzahl darstellt. In Zeile 21 wird die Funktion `T` aufgerufen, wobei die Rohdaten als Ganzzahl das übergebene Argument sind. In Zeile 22 wird eine Sekunde gewartet. Dies ist das Zeitintervall zwischen zwei Temperatur-Ausgaben.

Zeile 24 wird nach dem Drücken von `Ctrl-c` ausgeführt und macht eine einfache Bildschirmausgabe. Danach endet das Programm.

Display am I²C Bus

Wir verwenden nun einen I²C Datenbus des Raspberry Pi (siehe Seite 21), um ein Display JOY-iT 16X2 LCD MODUL anzusteuern, welches zwei Textzeilen mit jeweils 16 Zeichen zeigt. Der Anschluss dieses Displays an den Raspberry Pi wird in der Anleitung des Herstellers erklärt.

Nun wollen wir sehen, welche Bus-Adresse das an Kanal 1 angeschlossene Gerät (unser Display) hat. Der Befehl

```
i2cdetect -y 1
```

gibt eine Tabelle aus, welche die Adresse enthält. Wenn außer dem Display kein anderes Gerät am Bus ist, wird die Adresse `27` ausgegeben. Es handelt sich um eine Hexadezimalzahl, die man auch `0x27` schreibt. Sie entspricht der Dezimalzahl $2 \times 16 + 7 = 39$.

Wir wechseln in das Verzeichnis für Pythonprogramme (siehe Seite 48)

```
cd /home/ahg/progs
```

laden ein komprimiertes Treiberpaket für unser Display herunter

```
wget http://tutorials-raspberrypi.de/wp-content/uploads/scripts/hd44780_i2c.zip
```

und entpacken das Treiberpaket

```
unzip hd44780_i2c.zip
```

Wir erhalten die Dateien `i2c_lib.py` und `lcddriver.py`. Das komprimierte Treiberpaket `hd44780_i2c.zip` kann danach gelöscht werden.

Falls der Befehl `i2cdetect -y 1` (siehe oben) *nicht* die Hexadezimalzahl `0x27` ausgab, muss die Zeile `ADDRESS = 0x27` in der Datei `lcddriver.py` angepasst werden. Wenn nur ein Display am I^2C Datenbus angeschlossen ist, sollte das nicht nötig sein.

Wir schreiben mit einem Editor folgendes Pythonprogramm und speichern es unter dem Namen `lcd_test.py` im Verzeichnis `/home/ahg/progs`

```
1 import lcddriver                  # lade Displaytreibermodul
2 dp = lcddriver.lcd()              # Instanz für das Display
3 z1 = "1234567890123456"           # Text für die 1. Zeile
4 z2 = " abcdefg ABCDEFG"           # Text für die 2. Zeile
5 dp.lcd_clear()                    # lösche Display-Zeilen
6 dp.lcd_display_string(z1, 1)      # schreibe Display-Zeile 1
7 dp.lcd_display_string(z2, 2)      # schreibe Display-Zeile 2
```

Dann führen wir es aus

```
python3 lcd_test.py
```

In Zeile 1 des Displays sollte die Zeichenkette erscheinen, die unter der Referenz `z1` gespeichert wurde und in Zeile 2 die Zeichenkette, die unter der Referenz `z2` gespeichert wurde. Vermutlich sind die Zeichen nicht gut lesbar, weil der Kontrast noch nicht richtig eingestellt wurde. Die Kontrasteinstellung erfolgt mit einem passenden Schraubendreher am Trimmpotentiometer auf der Rückseite des Displays.

Im folgenden Beispiel werden Datum und Uhrzeit, sowie der Name des Rechners im Netzwerk (hostname) auf dem Display angezeigt. Die Anzeige erfolgt mit einem Pythonprogramm, das von einem Cronjob (siehe Seite 120) minütlich aufgerufen wird. Der Cronjob wird eingerichtet, indem wir mit dem Befehl

```
EDITOR=nano crontab -e
```

die Crontab des Nutzers `ahg` im Editor `nano` öffnen und am Ende die Zeile

```
* * * * * python3 /home/ahg/progs/lcd_clock.py >/dev/null 2>&1
```

anfügen. Die fünf Sternchen bedeuten, dass der Cronjob in jeder Minute einmal ausgeführt wird, wenn der Rechner läuft. Der Cronjob besteht in der Ausführung des Pythonprogramms `lcd_clock.py`, welches im Verzeichnis `/home/ahg/progs/` liegt. Die Zeichenfolge `>/dev/null 2>&1` bewirkt, dass Ausgaben dess Programms `lcd_clock.py` über den Standardausgabekanal (1) und den Standardfehlerkanal (2) verworfen werden.

Nach der eingefügten Zeile muss ein Zeilenumbruch folgen, der einfach durch Drücken der ENTER Taste gebildet wird, damit am Ende der Crontab eine leere Zeile steht. Nach dem Speichern der geänderten Datei (CTRL-o) verlässt man `nano` wieder (CTRL-x).

Nun schreiben wir das Pythonprogramm `lcd_clock.py`

```
# program lcd_clock.py
import lcddriver
from datetime import datetime
from socket import gethostname
dp = lcddriver.lcd()
t = datetime.now().strftime("%d.%m.%Y %H:%M")
u = gethostname()
dp.lcd_display_string(t, 1)
dp.lcd_display_string(u, 2)
```

und speichern es im Verzeichnis `/home/ahg/progs/`. Nach einem Neustart des Rechners sollte das Display Datum, Uhrzeit und Rechnername anzeigen.

8.2.11 Nutzer-Schnittstellen: simple-term-menu, urwid, curses

Eine bequeme Schnittstelle für den Nutzer kann aus einem einfachen Auswahlmenü bestehen und dann ist das Pythonmodul `simple-term-menu` geeignet. Vielfältiger und schöner, aber auch aufwändiger ist die Gestaltung mit der Python-Bibliothek `urwid`.

Einfaches Menu mit simple-term-menu

Das Modul `simple-term-menu` von Ingo Meyer kann in einem virtual environment installiert werden (siehe 262) oder für den aktuellen Nutzer als Teil von Python im Betriebssystem (siehe Seite 79) mit

```
pip3 install --user simple-term-menu --break-system-packages
```

Es ermöglicht ein einfaches Auswahlmenü, wie folgendes Beispiel zeigt. Die Auswahl erfolgt mit den Pfeiltasten ↓ und ↑ und der ENTER-Taste.

```
from simple_term_menu import TerminalMenu
options = [
"entry 1",
"entry 2",
]
def f_a():
    print("Decision: entry 1")
def f_b():
    print("Decision: entry 2")
flist = [f_a, f_b]
index = TerminalMenu(options).show()
flist[index]()
```

Die Liste der Optionen kann natürlich verlängert werden und die definierten Funktionen sollten vielleicht etwas sinnvolleres tun als in diesem einfachen Beispiel. Die Zuordnung einer Funktion zu einer ausgewählten Option wurde im Abschnitt „Auswahl aus vielen Möglichkeiten“ ab Seite 275 erklärt. Da `index` hier eine Referenz auf eine natürliche Zahl ist, kann `flist` als Liste, statt als Dictionary, definiert werden.

Terminal User Interface mit urwid

Die Python-Bibliothek `urwid` nutzt die Bibliothek `ncurses` des Betriebssystems, um ein Terminal User Interface (TUI) bereit zu stellen. Das Modul `curses` stellt ebenfalls eine Python-Schnittstelle (wrapper) für die in C geschriebene Bibliothek `ncurses` dar. Es ist daher eine Alternative zu `urwid`.

`urwid` verwendet graphische Steuerelemente, die *widgets* heißen. Sie zeigen etwas, zum Beispiel Text oder ein Bild, oder reagieren auf Eingaben des Nutzers. Durch eine Zusammenstellung von widgets kann man in einem Pythonprogramm mit `urwid` eine ansprechende graphische Darstellung im Terminal erreichen. Man darf aber nicht die gleiche Funktionalität wie bei Tkinter erwarten, was eine graphische Benutzeroberfläche (GUI) erfordert. Man installiert `urwid` mit

```
sudo apt install python3-urwid
```

urwid-Beispiel: Hello World!

Das folgende Programm zeigt ein einfaches Textfeld bis zum Abbruch durch Tastendruck.

```
1  # uw_1.py : Example program 1 for urwid
2  import urwid
3
4  # Funktion, die Eingabe (Tastendruck) verarbeitet
5  def fct(key):
6      if key in ("esc", "q","Q"):     # Tastendruck
7          raise urwid.ExitMainLoop() # exit exception
8
9  # Palette mit (Farbsatz, Vorder-, Hintergrundfarbe)
10 # mehrere Tupel, aber nur wenige verwendbare Farben
11 p = [ ("p1", "white", "black"),
12       ("p2", "light gray", "black"),
13       ("p3", "light red", "black"),
14       ("p4", "dark red", "dark green"),
15       ("p5", "yellow", "dark blue"),
16     ]
17
18 # Textwidget zeigt Text, mit Farbsatz durch AttrMAp
19 txt = urwid.Text("Hello World")
20 txt = urwid.AttrMap(txt, "p3")
```

```
21
22 # Fillerwidget (positioniert txt, füllt oben/unten)
23 fil = urwid.Filler(txt,"middle") # "top", "bottom"
24
25 # Endlosschleife für topmost widget (box widget)
26 # Palette p mit Farbsätzen, Eingabeverarbeitung fct
27 loop = urwid.MainLoop(fil, p, unhandled_input=fct)
28 loop.run() # Abbruch mit Esc, q oder Q (Shift-q)
```

Hier werden widgets vom Typ `Text` und `Filler` verwendet.

Um Farben in widgets verwenden zu können, wird eine oder mehrere Farbpaletten definiert. Eine Farbpalette (hier `p`) enthält Farbsätze. Jeder Farbsatz ist ein Tupel aus dem Namen des Farbsatzes, der Vordergrundfarbe und der Hintergrundfarbe.

`Text` stellt zeigt Text mit einer bestimmbaren Spaltenbreite (hier die ganze Bildschirmbreite) und kümmert sich um die Zeilenumbrüche. Da die Anzahl der dargestellten Zeilen automatisch an den Text angepasst wird, nennt man es ein „flow widget“.

Die Eigenschaften eines widgets (hier `txt`) können durch `AttrMap` (eine sogenannte decoration) verändert werden. Hier wird der Farbsatz `p3` angewandt.

`Filler` fügt Leerzeilen vor und nach einem anderen widget ein, sodass das andere widget oben, mittig oder unten steht (hier mittig). Da ein bestimmter Bildschirmbereich ausgefüllt wird (box = Kasten), nennt man es ein „box widget“.

Mit `MainLoop` wird eine Instanz erzeugt (hier `loop`), welche die Abbildung eines box widgets auf dem Bildschirm vorbereitet. Dieses übergeordnete widget (topmost widget, hier `fil`) kann andere widgets enthalten (hier `txt`). Außerdem wird festgelegt, welche Farbpalette verwendet werden soll (hier `p`).

Die mit `MainLoop` erzeugte Instanz (`loop`) hat die Methode `run()`, welche die bereits vorbereitete Abbildung auf dem Bildschirm in einer Endlosschleife durchführt.

In obigem Beispiel gibt es kein widget, das auf Nutzereingaben reagiert. Für diesen Fall wurde bei der Erzeugung der Instanz `loop` mit `unhandled_input=fct` festgelegt, dass eine Funktion `fct` aufgerufen wird, welche als Argument die Nutzereingabe erhält.

In der Funktion `fct` wird geprüft, ob die Escape-Taste `Esc`, oder die Taste `q` oder `Q` (`Shift-q`) gedrückt wurde. Wird eine andere Taste als `Esc`, `q` oder `Q` gedrückt, macht die Funktion `fct` nichts und die Endlosschleife läuft weiter.

Wenn die Taste `Esc`, `q` oder `Q` gedrückt wird, dann wird in der Funktion `fct` die exception (Ausnahme, siehe Seite 277) `ExitMainLoop` geworfen. Diese sorgt dafür, dass die Endlosschleife verlassen wird. Das Programm endet.

urwid-Beispiel: Auswahl-Menü für Internet-Radio

Das folgende Programm zeigt ein einfaches Menü, das die Auswahl eines Internet-Radiosenders mit Pfeiltasten und ENTER oder mit der Maus (sofern installiert) ermöglicht. Der gewählte Sender wird mit dem Mediaplayer MPlayer (siehe Seite 186) gespielt. Die Tastatursteuerung des MPlayers ist anwendbar. Abbruch durch `q`.

```
# uw_2.py : Example program for urwid (Internet-Radio)
import urwid as u
from subprocess import run

# Dictionary mit Menuepunkten und run-Listen
d = {"EXIT" : "exit",

"Audio Noir (USA), Drama (englisch),\
 48 kbit/s" : \
["mplayer", "-cache", "96","-cache-min", "50",\
 "-ao", "alsa", "http://198.245.61.123:8000/noir/"],

"Hörspielprojekt - Community für Hörspielmacher,\
 128 kbit/s" : \
["mplayer", "-cache", "256","-cache-min", "50",\
 "-ao", "alsa", "http://stream.laut.fm:80/hoerspiel"],

"Radio Waltl, freies Talk-Radio,\
 128 kbit/s" : \
["mplayer", "-cache", "256","-cache-min", "50",\
 "-ao", "alsa", "http://live.datamatix.at:8024"],

}
k=list(d.keys())
v=list(d.values())

def menu():
   o = [u.Text("\nWähle einen Sender:\n")]
   for x in k:
      b = u.Button(x)
      u.connect_signal(b, 'click', item)
      o.append(u.AttrMap(b,None,focus_map='reversed'))
   return u.ListBox(u.SimpleFocusListWalker(o))

# def item(button, choice):
def item(button):
   global n, s ; s = v[0]
   for x in range(1,len(d)):
      if button.get_label()==k[x]:
         s = k[x]
         n = v[x]
   raise u.ExitMainLoop()

```

```
44 p = [('reversed','standout','')]
45 u.MainLoop( menu(), palette=p ).run()
46
47 run(["clear"]) # Bildschirm löschen
48 if s==v[0]: print("Kein Sender gewählt.")
49 else:
50    print("\nInternet-Radio (q quit, 9 leiser 0 lauter)")
51    print("Sender:", s)
52    run(n)
```

Information zu `urwid` erhält man auf der Webseite http://urwid.org/ .

Ein Spiel mit urvid: usolitaire

Wenn man `urwid` und `fbterm` installiert hat und eine Maus angeschlossen ist, kann man eine einfache Version des Kartenspiels Klondike (Solitaire) spielen. Es wird mit

```
pip3 install --user usolitaire
```

installiert und mit

```
usolitaire
```

gestartet. Information dazu findet man auf der GitHub Webseite des Autors Elias Dorneles, https://github.com/eliasdorneles/usolitaire . Leider werden die Spielfarben mit dem Zeichensatz der Konsole nicht richtig angezeigt. Man muss daher das graphische Terminal `fbterm` verwenden, siehe Abschnitt 3.2.3 auf Seite 86.

Internet-Radio mit curses

Für das Betriebssystem Unix wurde die Programmbibliothek curses in der Programmiersprache C geschrieben, um zeichenorientierte Benutzerschnittstellen (text user interfaces) zu erzeugen. Eine neue Version für Linux ist ncurses. Das Pythonmodul namens curses ermöglicht die Nutzung von curses in einem Pythonprogramm für Linux.

Das folgende Programm zeigt ein Menü, das durch einfachen Tastendruck bedient wird (ohne nachfolgendes ENTER). Abhängig vom Tastendruck wird ein Steuerungsbefehl an das rc interface des Mediaplayers VLC (siehe Seite 193) übertragen. Für diese Übertragung wird das Modul pexpect eingesetzt. So entsteht ein Internet-Radio mit drei Sendern und Lautstärkeregelung.

```
1 # cursed-radio.py (internet radio with button controls)
2 import curses, pexpect
3
4 s1="https://dispatcher.rndfnk.com/br/br24/live/mp3/mid"
5 t1="BR24 Nachrichten, german (128 kbit/s)"
6 s2="https://s5.radio.co/sca4082ebb/listen"
```

```
7  t2="Kontrafunk, german (128 kbit/s)      "
8  s3="http://tunein.streamguys1.com/foxnews"
9  t3="FOX News, english (128 kbit/s)       "
10
11 def main(stdscreen):
12 vlc=pexpect.spawn("vlc -I rc")
13 curses.curs_set(False)
14 stdscreen.addstr(1,1,"Radio (control via buttons)")
15 stdscreen.addstr(3,1,"1  "+t1)
16 stdscreen.addstr(4,1,"2  "+t2)
17 stdscreen.addstr(5,1,"3  "+t3)
18 stdscreen.addstr(7,1,"up / down arrows control volume")
19 stdscreen.addstr(8,1,"q  to quit")
20 stdscreen.refresh()
21
22 while(True): # waits for key
23 char = stdscreen.getch()
24 if char == ord("q"):
25 vlc.sendline("quit")
26 break
27 elif char == curses.KEY_DOWN:
28 vlc.sendline("voldown")
29 elif char == curses.KEY_UP:
30 vlc.sendline("volup")
31 elif char == 49:    # 49 is ASCII for 1
32 vlc.sendline("quit")
33 stdscreen.addstr(10,1,"Now playing: "+t1)
34 vlc=pexpect.spawn("vlc -I rc "+s1)
35 elif char == 50:    # 50 is ASCII for 2
36 vlc.sendline("quit")
37 stdscreen.addstr(10,1,"Now playing: "+t2)
38 vlc=pexpect.spawn("vlc -I rc "+s2)
39 elif char == 51:    # 51 is ASCII for 3
40 vlc.sendline("quit")
41 stdscreen.addstr(10,1,"Now playing: "+t3)
42 vlc=pexpect.spawn("vlc -I rc "+s3)
43
44 curses.wrapper(main)
```

Da `curses` den Bildschirm belegt, kann man so nicht Internet-Fernsehen steuern. Das Modul `pexpect` muss vorher mit dem Befehl

```
sudo apt install python3-pexpect
```

installiert werden. Die Funktionen von `curses` verändern die Einstellungen des Terminals

und daher besteht die Gefahr, dass bei Programmfehlern ein ungeordneter Zustand des Terminals verbleibt. In Python kann man diese Gefahr bannen, indem man `curses` mit einer wrapper function aufruft, wie in obigem Beispiel gezeigt. Sie räumt nach den `curses`-Funktionen auf, auch im Fehlerfall.

Der Mauszeiger (cursor) wird mit dem Befehl `curses.curs_set(False)` ausgeblendet. Die Methode `addstr()` schreibt eine Zeichenkette auf den Bildschirm, beginnend an einer Position, deren y-Koordinate als Spalte und deren x-Koordinate als Zeile angegeben werden. Man beachte, dass hier die y-Koordinate vor der x-Koordinate steht.

8.2.12 PyGame

Die Programmbibliothek `PyGame` kann zur Entwicklung von Spielen dienen. Da sie effektiv (hardware accelerated) mit der framebuffer console (fbcon) arbeiten kann, ermöglicht sie graphische Ausgaben auch ohne X, und außerdem die Interaktion mit dem Nutzer über verschiedene Eingabegeräte. Sie ist daher ein interessantes Werkzeug für die Programmierung unter Linux ohne X. Man installiert sie mit

```
sudo apt install python3-pygame
```

und erhält `PyGame` mit *full image support*, sodass man beispielsweise Bilddateien mit den Formaten JPG oder PNG laden kann. Installiert man `PyGame` dagegen über `pip3`, erhält man normalerweise eine Version, welche nur unkomprimierte Bilddateien mit dem Format BMP laden kann.

PyGame-Methoden

Wichtige Methoden der obersten PyGame Stufe `pygame` sind unter anderem

`pygame.init()`	Kurzbefehl zur Initialisierung der wichtigsten PyGame Module
`pygame.quit()`	Rücksetzung aller PyGame Module in den Ruhezustand

Wichtige Methoden des PyGame Moduls `pygame.display` sind unter anderem

`pygame.display.init()`	initialisiert `pygame.display`, wenn `pygame.init()` fehlt
`pygame.display.flip()`	erneute Anzeige der gesamten PyGame Fläche
`pygame.display.update()`	wie `pygame.display.flip()`, aber Teilerneuerung möglich

Wichtige Methoden des PyGame Moduls `pygame.draw` sind unter anderem

`pygame.draw.rect()`	zeichnet Rechteck
`pygame.draw.circle()`	zeichnet Kreis
`pygame.draw.line()`	zeichnet Strecke, ohne Glättung

Beispiele für Methoden des PyGame Moduls `pygame.image` sind:

`so = pygame.image.load("bld.png")`	lädt Bild aus Datei, liefert pygame.Surface Objekt
`pygame.image.save(so,"bld.jpg")`	speichert Surface Objekt, Dateiendung → Format

Ein pygame.Surface Objekt `so` repräsentiert ein Bild, Beispiele für Methoden:

`bw = so.get_width()`	liefert Bildweite `bw` (in Pixeln) als Ganzzahl
`bh = so.get_height()`	liefert Bildhöhe `bh` (in Pixeln) als Ganzzahl
`bs = so.get_bytesize()`	liefert Anzahl der Bytes pro Pixel `bs` als Ganzzahl

PyGame-Namen für Farben und Tasten

PyGame kennt einige benannte Farben (named colors), die man in der PyGame Dokumentation findet. Unter anderem gibt es (RGB-Wert in Klammern):
antiquewhite (250, 235, 215), aquamarine (127, 255, 212), azure (240, 255, 255), black (0, 0, 0), blue (0, 0, 255), blueviolet (138, 43, 226), brown (165, 42, 42), burlywood (222, 184, 135), coral (255, 127, 80), cornflowerblue (100, 149, 237), cyan (0, 255, 255), darkblue (0, 0, 139), darkgoldenrod (184,134,11), darkgray (169,169,169), darkgreen (0, 100, 0), darkorange (255, 140, 0), darkred (139, 0, 0), darkviolet (148, 0, 211), forestgreen (34, 139, 34), gold (255, 215, 0), goldenrod (218, 165, 32), gray (128, 128, 128), greenyellow (173, 255, 47), lavender (230, 230, 250), lightblue (173, 216, 230), lightgoldenrod (238, 221, 130), lightgray (211, 211, 211), lightgreen (144, 238, 144), lightpink (255, 182, 193), lightyellow (255, 255, 224), magenta (255, 0, 255), midnightblue (25, 25, 112), navyblue (0, 0, 128), orange (255, 165, 0), orangered (255, 69, 0), pink (255, 192, 203), plum (221, 160, 221), purple 160, 32, 240), red (255, 0, 0), sandybrown (244, 164, 96), sienna (160, 82, 45), skyblue (135, 206, 235), springgreen (0, 255, 127), tan (210, 180, 140), tomato (255, 99, 71), violet (238, 130, 238), violetred (208, 32, 144), wheat (245, 222, 179), white (255, 255, 255), yellow (255, 255, 0), yellowgreen (154, 205, 50).

Alternativ kann man eine Farbe im RGB-Format definieren, zum Beispiel für Rot mit `textcolor = (255, 0, 0)` .

Man kan auf das Drücken einer Taste reagieren. Die PyGame Namen der Pfeiltasten sind `K_LEFT`, `K_RIGHT`, `K_UP` und `K_DOWN`. Die Namen der Leertaste, der Eingabetaste, der Tabulatortaste, der Backspace-Taste und der Escape-Taste sind `K_SPACE`, `K_RETURN`, `K_TAB`, `K_BACKSPACE` und `K_ESCAPE`. Die Taste für den Buchstaben a ist `K_a` und für die Ziffer 0 ist `K_0`. Die anderen Buchstaben und Ziffern funktionieren ebenso. Linke und rechte Umschalttasten (Shift-Taste) sind `K_LSHIFT` und `K_RSHIFT`. Die Insert-Taste (Einfügen) und die Delete-Taste (Entfernen) sind `K_INSERT` und `K_DELETE`.

Die Größe (size) der pygame.Surface

In einem Betriebssystem ohne Fenster nutzt PyGame eine rechteckige Teilfläche des Bildschirms des Computermonitors. Diese Teilfläche hat in der Literatur leider verschiedene Namen, aber ihr Datentyp ist `pygame.Surface` und wenn wir genau sein wollen, nennen wir sie Surface und als kurze Referenz verwenden wir `pgs` oder `s`.

Zunächst muss die Surface eingerichtet werden, indem sie initialisiert wird und die Betriebsart (der Modus) festgelegt wird. Die Initialisierung erfolgt der Surface durch den Befehl `pygame.display.init()` oder den übergeordneten Befehl `pygame.init()`. Der Modus wird danach mit dem Befehl `pygame.display.set_mode()` festgelegt.

Der Modus-Befehl hat optionale Parameter, die durch Kommata getrennt werden. Man kann die Größe (size) der Surface durch ein Paar (Breite, Höhe) in Pixeln angeben. Ohne diese Angabe wird die Größe der Surface gleich der Größe des ganzen Bildschirms gesetzt. Der Parameter `flags` wird automatisch gesetzt (`flags=pygame.NOFRAME`).

Es gibt verschiedene Methoden, um die Größe der Surface zu erfragen, wie folgendes Beispiel zeigt.

```
1  #!/usr/bin/env python3
2  # pygame_surface.py
3  import pygame
4
5  # initialize PyGame, includes pygame.display.init()
6  pygame.init()
7  # default size of the PyGame Surface = whole screen
8  size1 = (pygame.display.Info().current_w,\
9           pygame.display.Info().current_h)
10 # make PyGame Surface, size (width,height) in pixels
11 pgs = pygame.display.set_mode( size=(600,400) )
12 # new size of the PyGame Surface < whole screen
13 pgs_w = pgs.get_width() ; pgs_h = pgs.get_height()
14 size2 = (pygame.display.Info().current_w,\
15          pygame.display.Info().current_h)
16 # uninitialize PyGame (automatically when program exits)
17 pygame.quit()
18
19 # show size of the whole screen of the computer monitor
20 print("   width of the whole screen:", size1[0])
21 print("  height of the whole screen:", size1[1])
22 # show size of PyGame Surface (area in the whole screen)
23 print(" width of the PyGame Surface:", pgs_w)
24 print("height of the PyGame Surface:", pgs_h)
25 # show again size of PyGame Surface (with other methods)
26 print(" width of the PyGame Surface:", size2[0])
27 print("height of the PyGame Surface:", size2[1])
```

In der Festlegung des Modus genügt `(600,400)` statt `size=(600,400)`.

Bildschirmausgabe von Bilddateien

Das folgende Programm zeigt ein Bild, das in der Bilddatei `/home/ahg/image/erde.png` gespeichert ist, auf dem Bildschirm, bis eine beliebige Taste gedrückt wird.

```
#!/usr/bin/env python3
# pygame-test.py - show image file
import pygame
from pygame.locals import *
from os import system
#
# Variable mit Pfad zu einer Bilddatei
datei="/home/ahg/image/erde.png"
#
# Initilisierung und Anzeige des Bilds
system("clear")
pygame.display.init()
pygame.mouse.set_visible(False)
b=pygame.image.load(datei)
s=pygame.display.set_mode(b.get_size())
s.blit(b,(0,0))
pygame.display.flip()
#
# Warten auf Tastendruck und dann Ende
waiting=True
while waiting:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            var="QUIT" ; waiting=False
        elif event.type == pygame.KEYDOWN:
            var="KEYDOWN" ; waiting=False
pygame.quit()
#
print("Ende durch",var)
```

Die Funktion `system` wird vom Modul `os` bereitgestellt. Der Aufruf `system("clear")` löscht den Bildschirm vor der Anzeige des Bildes.

Die Erstellung einer Graphik durch `pygame` erfolgt zunächst im Arbeitsspeicher. Dort werden die Bildanteile auf einem Raster, das der räumlichen Auflösung des Bildschirms entspricht, übereinandergelegt. Diesen Vorhang nennt man „blitten“. Er geschieht in obigem Programm mit dem Befehl `s.blit(imgSurf,(0,0))`. Vom Arbeitsspeicher wird der gesamte von `pygame` genutzte Bildschirm mit `pygame.display.flip()` in den Grafikspeicher übertragen und damit angezeigt.

Abspiel einer MP3-Datei

PyGame kann Audiodateien im Format WAV oder MP3 abspielen:

```
1  #!/usr/bin/python3
2  # pygame-music.py - pPyGame spielt  eine MP3-Audiodatei
3  from pygame import mixer
4  datei="/home/ahg/audio/musi.mp3"
5  mixer.init()
6  print("Abspiel der Audiodatei", datei)
7  v = input("Lautststärke (0.0 - 1.0): ")
8  v = float(v)
9  v = min(v, 1.0)
10 v = max(v, 0.0)
11 mixer.music.set_volume(v)
12 mixer.music.load(datei)
13 mixer.music.play()
14 r=input("ENTER drücken zum Beenden")
```

Menü mit `pygame-menu`

Wir gehen davon aus, dass ein virtual environment namens `mypy` geschaffen wurde (siehe Seite 262), gehen in das entsprechende Verzeichnis, aktivieren unser virtual environment und installieren das Modul `pygame-menu` über `pip3`.

```
cd ~/progs/mypy
source bin/activate
python -m pip install pygame-menu
```

Da das virtual environment nur die wichtigsten Pythonmodule enthält, nicht aber das hier benötigte `pygame`, wird `pygame` automatisch ebenfalls über über `pip3` installiert.

Nun schreiben wir in `mypy` die folgende Pythondatei namens `pgmenu.py`.

```
1  #!/usr/bin/python3
2  # pgmenu.py
3  import pygame, pygame_menu
4
5  def set_vol(value, vol):
6      pygame.mixer.music.set_volume(vol)
7
8  def love():
9      f = "/home/ahg/audio/abba_peopleneedlove.mp3"
10     pygame.mixer.music.load(f)
11     pygame.mixer.music.play()
12
```

```
13 def ring():
14     f = "/home/ahg/audio/abba_ringring.mp3"
15     pygame.mixer.music.load(f)
16     pygame.mixer.music.play()
17 
18 def town():
19     f = "/home/ahg/audio/abba_anothertown.mp3"
20     pygame.mixer.music.load(f)
21     pygame.mixer.music.play()
22 
23 pygame.mixer.init()
24 pygame.mixer.music.set_volume(0.2)
25 
26 pygame.init()
27 screen = pygame.display.set_mode()
28 menu = pygame_menu.Menu(title="ABBA",
29 width=600, height=400, mouse_visible=False,
30 theme=pygame_menu.themes.THEME_SOLARIZED)
31 # more Themes: THEME_DEFAULT THEME_BLUE THEME_GREEN
32 menu.add.selector("Lautstärke: ", [("leise",0.2),
33 ("mittel",0.5),("laut",1.0) ], onchange=set_vol)
34 menu.add.button("People Need Love", love)
35 menu.add.button("Ring Ring", ring)
36 menu.add.button("Another Town, Another Train", town)
37 menu.add.button('ENDE', pygame_menu.events.EXIT)
38 menu.mainloop(screen)
```

Wir brauchen hier natürlich die drei Audiodateien
`/home/ahg/audio/abba_peopleneedlove.mp3`
`/home/ahg/audio/abba_ringring.mp3`
`/home/ahg/audio/abba_anothertown.mp3`
(oder drei andere) und können dann mit dem Befehl

```
python pgmenu.py
```

ein schönes Menü kochen. Mit Pfeiltasten und ENTER können wir ein Musikstück aussuchen und abspielen oder die Lautstärke ändern. Mit dem Befehl

```
deactivate
```

beenden wir das virtual environment und kehren ins normale Betriebssystem zurück.

Bildschirmausgabe von Videodateien

Das Modul *MoviePy* kann nicht nur Videos verschiedener Formate abspielen, sondern auch bearbeiten. Wie auf Seite 79 erläutert, installiert man es mit

```
pip3 install --user moviepy --break-system-packages
```

Zum Abspielen eines Video `/home/ahg/video/film.mp4` genügt folgendes Programm.

```
#!/usr/bin/python3
# pgvideo.py- plays a video file
import os
from moviepy.editor import *
import pygame
h = os.path.expanduser("~") # home directory
v = h+"/video/film.mp4"
if os.path.exists(v):
    VideoFileClip(v).preview()
else:
    print("File not found:",v)
pygame.quit()
```

Der Funktionsaufruf `os.path.expanduser(-")` gibt das Heimatverzeichnis des Nutzers zurück. Natürlich kann man das Programm mit den Möglichkeiten von `PyGame` erweitern. Ebenso werden Einzelbilder und Animationen im GIF-Format angezeigt, wenn ImageMagick installiert wurde. Auch Bilder im JPEG-Format werden angezeigt. Voraussetzung ist aber, dass die Bilder nicht zu groß sind (in Pixeln gemessen). Beim Format PNG gab es eine Fehlermeldung, die ich nicht weiter untersucht habe. Benutzt man obiges Programm zur Anzeige eines Bildes, sollte man die Programmausführung am Ende anhalten, bis eine Nutzereingabe erfolgt, damit genug Zeit zum Betrachten bleibt.

Textausgabe

PyGame kann eine Textzeile mithilfe der fonts (Zeichensätze), die dem Betriebssystem zur Verfügung stehen, wie ein Bild auf dem Bildschirm ausgeben. Dies geschieht in folgendem Programm.

```
#!/usr/bin/python3
# pygame-text.py - show text on screen
import pygame
from pygame.locals import *
#
# Text definition
list_of_lines = [
"Hallo Jena!",
"Hallo Welt!",
]
textcolor = pygame.Color("gold") # named color
# https://www.pygame.org/docs/ref/color_list.html
#
```

```
14 # Text output function
15 def text_output():
16     font = pygame.font.SysFont("notosansmono",25)
17     x_pos = 80 ; y_pos = 80    # Anfangsposition
18     for line in list_of_lines:
19         bild=font.render(line, True, textcolor)
20 # parameter True:  use antialiasing  (looks better)
21         screen.blit(bild,(x_pos, y_pos))
22         y_pos = y_pos + 30
23 #
24 # Inition of Pygame, text output and display
25 pygame.init()
26 pygame.mouse.set_visible(False)
27 screen = pygame.display.set_mode()
28 # without parameters:  use current screen resolution
29 backcolor = pygame.Color("darkgray") # named color
30 screen.fill(backcolor)
31 text_output()
32 pygame.display.flip()
33 #
34 # Waiting for a key press
35 waiting=True
36 while waiting:
37     for event in pygame.event.get():
38         if event.type == pygame.KEYDOWN:
39             waiting=False
40 #
41 # End
42 # print(pygame.font.get_fonts()) # available fonts
43 print(pygame.display.set_mode())
44 pygame.quit()
```

In Zeilen 5 bis 9 wird eine Liste erzeugt, deren Elemente die einzelnen Zeilen des Texts sind. Anschließend wird in Zeile 10 die Farbe festgelegt (die benannte Farbe `gold`), in welcher der Text erscheinen soll.

Die Befehle zur Textausgabe sind in der Funktion `text_output` zusammengefasst, die in Zeilen 13 bis 22 definiert wird. In Zeile 15 wird der font und die Zeichengröße (genauer: die Kegelhöhe) festgelegt. Die Anfangsposition der vom Text eingenommenen Fläche (linke, obere Ecke) wird in Zeile 16 festgelegt. In Zeilen 17 bis 21 werden die Textzeilen zu Bildelementen gemacht ("gerendert") und auf einen Speicherbereich für die Bildfläche übertragen (aber noch nicht angezeigt). Der in Zeile 21 gewählte Abstand der Textzeilen (30 Pixel) ist etwas größer als die Kegelhöhe (25 Pixel).

In den Zeilen 23 bis 27 wird PyGame initialisiert (Zeile 24), die Maus (sofern vorhanden) unsichtbar gemacht (Zeile 25) und die Bildfläche initialisiert (Zeile 26), wobei

hier (ohne Parameterangabe) Breite und Höhe (als Zahl der Pixel) vom Betriebssystem übernommen werden. In den Zeilen 28 und 29 wird die Hintergrundfarbe des Bildschirms festgelegt. In Zeile 31 wird die Funktion `text_output` (siehe oben) aufgerufen. Schließlich wird in Zeile 31 der Speicherbereich für die Bildfläche auf den Bildschirm übertragen und damit angezeigt.

In Zeilen 33 bis 38 wird eine unbegrenzte Schleife ausgeführt, in der auf einen beliebigen Tastendruck gewartet wird. Wenn er erfolgt, wird die Schleife verlassen.

Zeilen 41 und 42 zeigen, wie Information ausgegeben werden kann, nämlich die im Betriebssystem verfügbaren Fonts (Zeile 41), sowie Breite und Höhe der Bildfläche in Pixeln (Zeile 42). Schließlich wird PyGame in Zeile 43 sauber beendet.

Ausgabe von Bild und Text (von einer Webseite geladen)

Die Webseite https://earthobservatory.nasa.gov/topic/image-of-the-day zeigt jeden Tag ein neues Bild vom NASA Earth Observatory. Im folgenden Programm zeigt PyGame das neueste Bild mit überlagerter Bildüberschrift.

```
1  #!/usr/bin/python3
2  # id.py shows the NASA Image of the Day
3  import os, pygame, pygame.locals, re, requests
4
5  # dowload website text from earth observatory server
6  u1 = "https://earthobservatory.nasa.gov/"
7  u2 = "topic/image-of-the-day"
8  try:
9      rsp = requests.get(u1+u2)
10     txt = rsp.text
11 except requests.exceptions.RequestException as e:
12     raise SystemExit(e)
13
14 # save URL of image file and the image caption text
15 rs = r'<img src = "([^"]+)" alt = "([^"]+)"'
16 mao = re.compile(rs).search(txt)
17 if mao:
18     img = mao.group(1)
19     alt = mao.group(2)
20 else: print("ERROR: image file not found!")
21
22 # dowload image and save it in a temporary file
23 try:
24     rsp = requests.get(img)
25 except requests.exceptions.RequestException as e:
26     raise SystemExit(e)
27 with open("nasa.tmp", "wb") as f:
```

```
28      f.write(rsp.content)
29
30  # display image from temporary file using PyGame
31  os.system("clear")
32  pygame.display.init()
33  pygame.mouse.set_visible(False)
34  b = pygame.image.load("nasa.tmp")
35  s = pygame.display.set_mode()
36  s.blit(b, (0,0))
37
38  # display caption alt of the image using PyGame
39  pygame.font.init()
40  font = pygame.font.SysFont("notosansmono",25)
41  bild = font.render(alt, True, pygame.Color("gold"))
42  s.blit(bild,(8,0))
43
44  # display image and caption, delete temporary file
45  pygame.display.flip()
46  os.remove("nasa.tmp")
47
48  # wait until any key is pressed and finish PyGame
49  waiting = True
50  while waiting:
51      for event in pygame.event.get():
52          if event.type == pygame.locals.QUIT or\
53          event.type == pygame.locals.KEYDOWN:
54              waiting = False
55  pygame.quit()
```

Das Herunterladen eines Bildes mit dem Modul `requests` wurde bereits oben dargestellt, siehe Seite 344. Dort wurde auch (ähnlich wie hier) der Textinhalt einer Webseite mithilfe eines Regulären Ausdrucks (siehe Seite 290) ausgewertet.

Bildschirmausgabe von Matplotlib-Graphiken

Mithilfe von PyGame kann man Matplotlib-Graphiken direkt (ohne den Umweg über eine Bilddatei) auf dem Bildschirm darstellen. Das folgende Programm plottet Datenpunkte in einem Liniendiagramm wie im Beispiel aus Abschnitt 8.2.3 (auf Seite 313). Eine Speicherung in einer Bilddatei erfolgt nicht, nur eine direkte Bildschirmausgabe.

```
1  #!/usr/bin/python3
2  # pygame-plot.py - show line plot
3  from matplotlib.backends.backend_agg import \
4      FigureCanvasAgg
```

```
5  import matplotlib.pyplot as plt
6  import pygame
7  from pygame.locals import *
8  from os import system
9  #
10 # Daten für x- und y-Koordinaten, und ihre Gestaltung
11 lx = [-2,-1,0,1,2]        # Liste der x-Koordinaten
12 ly = [4.8,2,1.5,5.5,3]    # Liste der y-Koordinaten
13 fmt='bo-' # Formatangabe für Datenpunkte und  Linie
14 #
15 # Gestaltung der Achsen und Erzeugung des Plots
16 fig = plt.figure(figsize=(6.4,4.8),dpi=100) # defaults
17 plt.xlabel('unabhängige Variable') # x-Beschriftung
18 plt.ylabel('abhängige Variable')   # y-Beschriftung
19 plt.axis([-3,3,0,6]) # Achsen: xmin,xmax,ymin,ymax
20 plt.plot(lx,ly,fmt)  # Plot-Erzeugung, keine Ausgabe
21 #
22 # Vorbereitung der Graphik für pygame
23 cv = fig.canvas.switch_backends(FigureCanvasAgg)
24 size = cv.get_width_height() ; cv.draw()
25 raw_data = cv.get_renderer().tostring_rgb()
26 #
27 # Anzeige der Graphik mithilfe von pygame
28 system("clear")
29 pygame.display.init()
30 pygame.mouse.set_visible(False)
31 pygame.display.set_mode(size)
32 s = pygame.display.get_surface()
33 i = pygame.image.fromstring(raw_data, size, "RGB")
34 s.blit(i, (0,0)) ; pygame.display.flip()
35 #
36 # Warten auf Tastendruck und dann Ende
37 waiting=True
38 while waiting:
39     for event in pygame.event.get():
40         if event.type == pygame.QUIT:
41             var="QUIT" ; waiting=False
42         elif event.type == pygame.KEYDOWN:
43             var="KEYDOWN" ; waiting=False
44 pygame.quit() ; print("Ende durch",var)
```

PyGame mit Eingaben über GPIO

Das folgende Beispiel geht davon aus, dass drei Taster angeschlossen sind, ein grüner (an GPIO 27), ein roter (an GPIO 22) und ein schwarzer (an GPIO 25). Auf dem Bildschirm wird ein blauer Kreis gezeichnet. Drücken des roten Tasters bewegt den Kreis nach links und Drücken des grünen Tasters bewegt den Kreis nach rechts. Durch Drücken des schwarzen Tasters wird das Programm beendet.

```
#!/usr/bin/python3
# pygame_gpio.py - pygame with gpio input
import pygame
from gpiozero import Button
from time import sleep

# Taster: grün (GPIO27), rot (GPIO22), schwarz (GPIO25)
b = 0.01 # Prellzeit in Sekunden
tst1 = Button(pin=27,pull_up=None,active_state=True,
bounce_time=b)
tst2 = Button(pin=22,pull_up=None,active_state=True,
bounce_time=b)
tst3 = Button(pin=25,pull_up=None,active_state=True,
bounce_time=b)

# Initialisierung
x = 200 ; y = 100 # Startkoordinaten (pixels)
pygame.init() ; pygame.mouse.set_visible(False)
screen = pygame.display.set_mode()

# Unendliche Schleife mit Abfrage der Taster
warten = True
while warten:
    # Bewegung nach links oder rechts mit den Tastern
    if tst2.is_pressed: x = x - 10
    if tst1.is_pressed: x = x + 10
    # Abbruch des Programms durch den schwarzen Taster
    if tst3.is_pressed: warten = False
    # Hintergrund einfärben und Kreis zeichnen
    screen.fill(pygame.Color("darkgray"))
    pygame.draw.circle(screen, pygame.Color("blue"),
(x,y), 20, 0)
    pygame.display.flip()
    # Abbruch an Grenzen
    if x <= 0 or x >= 300: warten = False
    # Pause in s, damit der Rechner anderes tun kann
```

```
37      sleep(0.1)
```

8.2.13 Sprachausgabe und Spracherkennung

Sprachausgabe (online)

Wenn eine Internetverbindung vorhanden ist, kann man mit der Pythonbibliothek `gTTS` (Google Text-to-Speech) die Umwandlung von Sprache in Text mit der Hilfe von Google durchführen. Man installiert gTTS durch

```
pip3 install --user gTTS
```

Folgendes Beispiel zeigt die Sprachausgabe. Ein englischer Text wird von gTTS in eine Audiodatei verwandelt, welche das MP3-Format hat. Sie wird anschließend mit dem VLC (Schnittstellenmodul `dummy`, siehe Seite 192) abgespielt.

```
1 # Programm textos.py Text -> Sprache (online)
2 from gtts import gTTS
3 from subprocess import run
4 txt_1 = "The quick brown fox jumps again."
5 spr_1 = "en"    # en = englisch, de = deutsch
6 dat_1 = "ts_1.mp3"
7 myobj = gTTS(text=txt_1,lang=spr_1,slow=False)
8 myobj.save(dat_1)
9 run( [ "cvlc", "--play-and-exit", dat_1 ] )
```

Ein Text sollte mit der richtigen Sprache gesprochen werden. Mit dem Befehl

```
gtts-cli --all | less
```

kann man sich im Terminal die verfügbaren Sprachen anzeigen lassen. Dazu gehören Arabisch (ar), Deutsch (de), Englisch (en), Französisch (fr), Latein (la), Chinesisch (Mandarin, zh) und viele andere.

Spracherkennung

Wir brauchen eine Verbindung zum Internet und die Hilfe vom *Google Speech Recognition service*. Zunächst installieren wir in üblicher Weise (`sudo apt-get install ...`, siehe Seite 77) die Pakete `python3-pyaudio` und `flac`. Dann installieren wir ein Python-Modul zur Spracherkennung

```
pip3 install --user SpeechRecognition
```

Nun schließen wir ein Mikrophon an (siehe Seite 72) und geben den Befehl

```
python3 -m speech_recognition
```

ein. Dann ignorieren wir einige Warnungen auf dem Bildschirm. Wenn die Aufforderung `Say something!` auf dem Bildschirm erscheint, sprechen wir laut und deutlich „I love my tea.“ ins Mikrophon und erhalten hoffentlich die Auskunft `You said I love my tea`.

Im folgenden Programm wollen wir eine Audiodatei mit deutscher Sprache in einen Text umwandeln und diesen auf dem Bildschirm ausgeben.

Die Audiodatei kann zum Beispiel mit einem USB-Mikrophon und dem Befehl `arecord` erzeugt werden, wie in Abschnitt 2.2.2 auf Seite 72 beschrieben.

```
1  # Python program spr.py transforms speech -> text
2  import speech_recognition as sr
3  r = sr.Recognizer() # create Recognizer instance
4  #  audio file formats: WAV, AIFF, AIFF-C or FLAC
5  # we use file spr_de.wav (with audio format WAV)
6  audat = sr.AudioFile("spr_de.wav")    # in german
7  with audat as source:  # transform audio file ->
8      audio = r.record(source) # AudioData instance
9  # languages: en-US (US english, default),
10 #             de-DE (german), zh-CN (chinese), ...
11 # send AudioData instance to Google Web Speech API
12 t = r.recognize_google(audio, language="de-DE")
13 print(t) # Ausgabe des erkannten deutschen Texts
```

8.2.14 Eigene Module

Python-Code , welcher in einer Datei mit der Endung `.py` gespeichert ist, ist ein *Modul*. Wir können ein eigenes Modul `mym` schreiben, welches zum Beispiel eine Funktion definiert, und es in einer Datei `mym.py` speichern.

Ebenso kann man Module, die nicht zum Standard der Pythoninstallation gehören, von anderen Quellen laden. Für sie gilt im folgenden das gleiche wie für eigene Module.

Das eigene Modul `mym` kann in ein anderes Modul mit dem Befehl `import mym` eingebunden werden, wenn der Pythoninterpreter die Datei `mym.py`findet. Der Pythoninterpreter sucht zunächst im Verzeichnis, aus dem das aktuelle Pythonprogramm gestartet wurde, dann in den Verzeichnissen, die in der Umgebungsvariablen des Betriebssystems PYTHONPATH aufgeführt werden. Wenn man diese Ungebungsvariable nicht selbst gesetzt hat, gibt es sie meistens nicht.

Gibt es die Umgebungsvariable PYTHONPATH nicht, wird in Verzeichnissen gesucht, welche das Betriebssystem vorgibt, zum Beispiel in `/usr/lib/python3.xx`, wobei `3.xx` für die installierte Pythonversion steht. Es gibt Verzeichnisse von Pythonmodulen, die allen Nutzern des Betriebssystems zur Verfügung stehen und solche, die nur einem Nutzer gehören (local). Im Terminal erfährt man mit dem Befehl

```
python3 -m site
```

in welchen Verzeichnissen Python sucht. In einem Pythonprogramm erhält man die Information mit folgendem kleinen Programm:

```
import sys
print(sys.path)
```

Das Betriebssystem Debian (und Abkömmlinge davon, wie unser Raspberry OS oder Ubuntu) sortiert Module, die zusätzlich zum Pythonstandard über die Paketverwaltung installiert werden, in Verzeichnisse die `dist-packages` im Namen enthalten. Dazu gehören auch Module, die zum Beispiel von `pip3`) installiert werden, wenn `pip3`) über die Paketverwaltung installiert wurde. Module (von Drittanbietern), die man selbst (manuell) installiert (zum Beispiel mit dem Befehl `python3 setup.py install` ...), kommen in Verzeichnisse die `site-packages` im Namen enthalten. Warnung: Verschiedene Verzeichnisse können Module gleichen Namens enthalten.

Hilfe zu einem Modul `X` erhält man im interaktiven Modus von Python mit dem Befehl

```
help(X)
```

und dies gilt auch für eigene Module, wenn sie mit `import X` geladen wurden. Es ist sinnvoll, eigene Module und Funktionen mit einem Kommentar zu versehen, der im interaktiven Modus von der Hilfe angezeigt wird. Es ist üblich, hierfür Docstrings zu verwenden, welche entweder durch drei einfache oder drei doppelte Anführungszeichen begonnen und beendet werden.

Als Beispiel schreiben wir das Modul `test01` und speichern es in Datei `test01.py`

```
1 ''' Modul für Belangloses '''
2
3 def sin(x):
4     """ Ausgabe des Produkts von 6 und x """
5     print(str(6*x))
```

und schreiben ein Hauptprogramm, welches wir in der Datei `test02.py` speichern.

```
1 ''' Belangloses Produkt '''
2
3 import test01
4 test01.sin(7)
```

Die Ausführung von `test02.py` ergibt `42` . Im interaktiven Modus von Python erzeugt, nach dem Laden des Moduls `test01`, der Hilferuf `help(test01)` die Ausgabe

```
Help on module test01:

NAME
    test01 - Modul für Belangloses

FUNCTIONS
    sin(x)
```

```
        Ausgabe des Produkts von 6 und x

FILE
    /home/ahg/python/test01.py
```

9 Anhang

9.1 LaTeX-Code für Zeichnungen

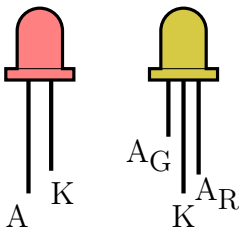
Farbcodierung von Widerständen mit vier Ringen

Abbildung 1.7 auf Seite 23 zeigt die Farbcodierung von Festwiderständen mit vier Ringen. Die Zeichnung wurde mit LaTeX und dem Paket `tikz` erstellt. Werte physikalischer Größen wurden mit dem Paket `siunitx` dargestellt.


```
% Präambel
\documentclass[a4paper,12pt,parskip]{scrartcl}
\usepackage[ngerman]{babel} \usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}   \pagestyle{empty}
\usepackage[output-decimal-marker={,}]{siunitx}   \usepackage{tikz}
\definecolor{gfy}{cmyk}{0,0.14,0.56,0.05}
% document-Umgebung
\begin{document}   \begin{tikzpicture}
\draw[thin,fill=lightgray] (-0.9,-0.3) rectangle (0.9,0.3) ;
\draw[ultra thick] (-2,0)--(-0.9,0) (0.9,0)--(2,0) ;
\fill[thin,fill=brown] (-0.7,-0.3) rectangle (-0.5,0.3) ;
\fill[thin,fill=black] (-0.3,-0.3) rectangle (-0.1,0.3) ;
\fill[thin,fill=red] (0.1,-0.3) rectangle (0.3,0.3) ;
\fill[thin,fill=gfy] (0.5,-0.3) rectangle (0.7,0.3) ;
\node at (0,-0.8) {\large\SI{1}{\kilo\ohm}} ;
\end{tikzpicture}   \end{document}
```

Anschlüsse (Anode und Kathode) von Leuchtdioden (LED)

Abbildung 1.10 auf Seite 27 zeigt die Anschlüsse von einer Standard-LED und einer Duo-LED (gemeinsame Kathode). Die Zeichnung wurde mit LaTeX und dem Paket `tikz` erstellt.




```
% Präambel
\documentclass[a4paper,12pt,parskip]{scrartcl}
\usepackage[ngerman]{babel} \usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}   \pagestyle{empty}
\usepackage{tikz}
```

```
% document-Umgebung
\begin{document}
\begin{tikzpicture}[scale=0.5]
\draw[fill=red!50!white,very thick] (6mm,0)--(6mm,10mm)
   arc[start angle=0,end angle=180,radius=6mm] --(-6mm,0)
   --(-9mm,0)--(-9mm,-3mm)--(9mm,-3mm)--(9mm,0)--cycle ;
\draw (-9mm,0)--(9mm,0) ;
\draw[ultra thick] (-3mm,-3mm)--(-3mm,-33mm) ;
\draw[ultra thick] (3mm,-3mm)--(3mm,-27mm) ;
\node at (-6mm,-39mm) {A} ;   \node at (6mm,-33mm) {K} ;
%
\pgfmathsetmacro{\dt}{38mm}
\draw[fill=yellow!80!black,very thick] (\dt+6mm,0)--(\dt+6mm,10mm)
   arc[start angle=0,end angle=180,radius=6mm] --(\dt+-6mm,0) --
   (\dt+-9mm,0)--(\dt+-9mm,-3mm)--(\dt+9mm,-3mm)--(\dt+9mm,0)--cycle ;
\draw (\dt+-9mm,0)--(\dt+9mm,0) ;
\draw[ultra thick] (\dt-4mm,-3mm)--(\dt-4mm,-18mm) ;
\draw[ultra thick] (\dt-0mm,-3mm)--(\dt-0mm,-33mm) ;
\draw[ultra thick] (\dt+4mm,-3mm)--(\dt+4mm,-28mm) ;
\node at (\dt+-9mm,-23mm) {A\textsubscript{\small G}} ;
\node at (\dt+0mm,-39mm) {K} ;
\node at (\dt+9mm,-33mm) {A\textsubscript{\small R}} ;
\end{tikzpicture}   \end{document}
```

Transistor-Anschlüsse

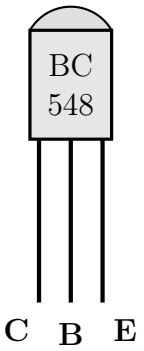


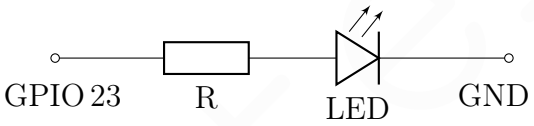
Abbildung 1.11 auf Seite 29 zeigt die Anschlüsse des Transistors BCE 548: **C**ollektor, **B**asis und **E**mitter. Die Zeichnung wurde mit LaTeX und dem Paket `tikz` erstellt.

```
% Präambel
\documentclass[a4paper,12pt,parskip]{scrartcl}
\usepackage[ngerman]{babel} \usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}   \pagestyle{empty}
\usepackage{tikz}
% document-Umgebung
\begin{document}
\begin{tikzpicture}[scale=0.6]
\fill[fill=gray!25!white] (9mm,12mm) arc [start angle=30,
   end angle=150, radius=10.392305mm] -- cycle ;
\draw[thick] (9mm,12mm) arc [start angle=30,
   end angle=150, radius=10.392305mm] ;
```

```
\draw[very thick,fill=gray!20!white]
   (-9mm,-12mm) rectangle (9mm,12mm) ;
\draw[ultra thick] (-7mm,-12mm) -- (-7mm,-48mm) ;
\draw[ultra thick] (0,-12mm) -- (0,-48mm) ;
\draw[ultra thick] (7mm,-12mm) -- (7mm,-48mm) ;
\node[align=center] at (0,0) {BC\\548} ;
\node at (-12mm,-54mm) {\textbf{C}} ;
\node at (0,-55mm) {\textbf{B}} ;
\node at (12mm,-54mm) {\textbf{E}} ;
\end{tikzpicture}   \end{document}
```

Schaltplan zur direkten LED-Ansteuerung durch einen GPIO

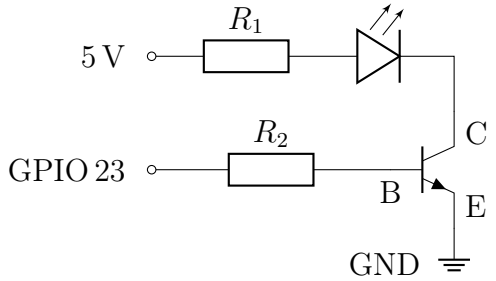
Abbildung 1.16 auf Seite 33 zeigt einen Schaltplan für die LED-Ansteuerung durch einen GPIO. Die Zeichnung wurde mit LaTeX und dem Paket `circuitikz` erstellt.




```
% Präambel
\documentclass[a4paper,12pt,parskip]{scrartcl}
\usepackage[ngerman]{babel} \usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}    \pagestyle{empty}
\usepackage[european,nooldvoltagedirection]{circuitikz}
% document-Umgebung
\begin{document}   \begin{circuitikz}
\draw (0,0) to[short,o-] (1,0) to[resistor,l_=R] (3,0)
   to[empty led,l_=LED] (5,0) to[short,-o] (6,0) ;
\node[below] at (0.1,-0.2) {GPIO\,23} ;
\node[below] at (5.8,-0.2) {GND} ;
\end{circuitikz}   \end{document}
```

Schaltplan zur LED-Ansteuerung von einem GPIO über einen npn-Transistor

Abbildung 1.18 auf Seite 34 zeigt einen Schaltplan für die LED-Ansteuerung über einen npn-Transistor mit Emitterschaltung. Die Zeichnung wurde mit LaTeX und dem Paket `circuitikz` erstellt. Die Darstellung der Werte physikalischer Größen erfolgte mit dem Paket `siunitx`.

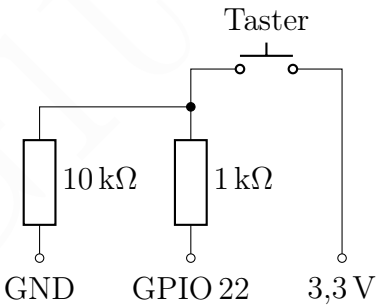



```
% Präambel
\documentclass[a4paper,12pt,parskip]{scrartcl}
\usepackage[ngerman]{babel} \usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}    \pagestyle{empty}
```

```
\usepackage[output-decimal-marker={,}]{siunitx}
\usepackage[siunitx,european,nooldvoltagedirection]{circuitikz}
% document-Umgebung
\begin{document}   \begin{circuitikz}
\draw (0,0) node[npn](npn) {}
   (npn.base) node[below] {B}
   (npn.collector) node[below right] {C}
   (npn.emitter) node[above right] {E} ;
\draw (-4,0) to[short,o-] (-4,0) to[resistor,l^=$R_2$] (npn.base);
\draw (-4,1.5) to[short,o-] (-3.5,1.5) to[resistor,l^=$R_1$]
   (-2,1.5) to[empty led,] (0,1.5) to (npn.collector);
\node[ground] at (npn.E) {};
\node[left] at (-4.2,1.5) {\SI{3,3}{\volt}} ;
\node[left] at (-4.2,0) {GPIO\,23} ;
\node[left] at (-0.3,-1.25) {GND} ;
\end{circuitikz}   \end{document}
```

Schaltplan zur Auslesung eines Tasters (Schließer) durch einen GPIO

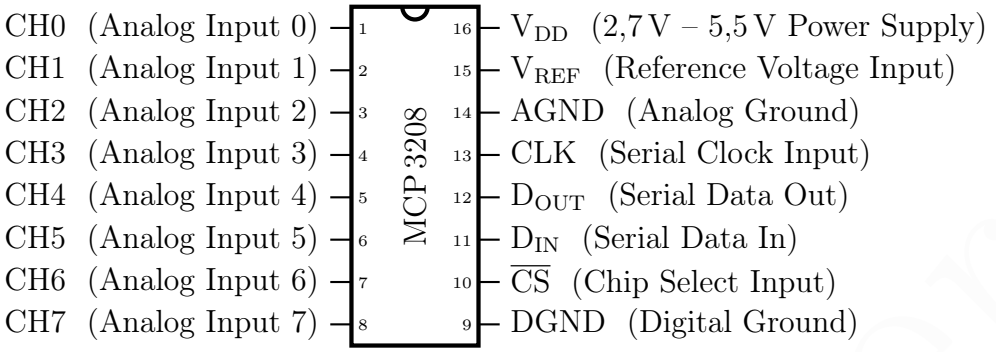
Abbildung 1.19 auf Seite 35 zeigt einen Schaltplan für die Auslesung eines Tasters durch einen GPIO. Die Zeichnung wurde mit LaTeX und dem Paket `circuitikz` erstellt. Die Darstellung der Werte physikalischer Größen erfolgte mit dem Paket `siunitx`.




```
% Präambel
\documentclass[a4paper,12pt,parskip]{scrartcl}
\usepackage[ngerman]{babel} \usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}    \pagestyle{empty}
\usepackage[output-decimal-marker={,}]{siunitx}
\usepackage[siunitx,european,nooldvoltagedirection]{circuitikz}
% document-Umgebung
\begin{document}   \begin{circuitikz}
\draw (0,0) to[resistor,l_=\SI{10}{\kilo\ohm},o-] (0,2)
   to[short,-*] (2,2) ;
\node[below] at (0,-0.1) {GND} ;
\draw (2,0) to[resistor,l_=\SI{1}{\kilo\ohm},o-] (2,2) ;
\node[below] at (2,-0.1) {GPIO\,22} ;
\draw (2,2) to (2,2.5) to[push button,l^=Taster] (4,2.5)
   to[short,-o] (4,0) ;
\node[below] at (4,-0.1) {\SI{3,3}{\volt}} ;
\end{circuitikz}   \end{document}
```

Anschlüsse (Pins) des MCP 3208

Abbildung 1.15 auf Seite 32 zeigt die Anschlüsse des A/D-Wandlers MCP 3208. Die Zeichnung wurde mit LaTeX und dem Paket `circuitikz` erstellt. Die Darstellung der Werte physikalischer Größen erfolgte mit dem Paket `siunitx`.




```
% Präambel
\documentclass[a4paper,12pt,parskip]{scrartcl}
\usepackage[ngerman]{babel} \usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}    \pagestyle{empty}
\usepackage[output-decimal-marker={,}]{siunitx}
\usepackage[siunitx,european,nooldvoltagedirection]{circuitikz}
% document-Umgebung
\begin{document}   \begin{circuitikz}
\draw[thick] (0,0) node[dipchip,
num pins=16] (IC){\rotatebox{90}{MCP\,3208}};
\node[left] at (IC.pin 1) {CH0~~(Analog Input 0)} ;
\node[left] at (IC.pin 2) {CH1~~(Analog Input 1)} ;
\node[left] at (IC.pin 3) {CH2~~(Analog Input 2)} ;
\node[left] at (IC.pin 4) {CH3~~(Analog Input 3)} ;
\node[left] at (IC.pin 5) {CH4~~(Analog Input 4)} ;
\node[left] at (IC.pin 6) {CH5~~(Analog Input 5)} ;
\node[left] at (IC.pin 7) {CH6~~(Analog Input 6)} ;
\node[left] at (IC.pin 8) {CH7~~(Analog Input 7)} ;
\node[right] at (IC.pin 9) {DGND~~(Digital Ground)} ;
\node[right] at (IC.pin 10)
{$\overline{\text{CS}}$~~(Chip Select Input)} ;
\node[right] at (IC.pin 11) {D${}_{\text{IN}}$~~(Serial Data In)} ;
\node[right] at (IC.pin 12) {D${}_{\text{OUT}}$~~(Serial Data Out)} ;
\node[right] at (IC.pin 13) {CLK~~(Serial Clock Input)} ;
\node[right] at (IC.pin 14) {AGND~~(Analog Ground)} ;
\node[right] at (IC.pin 15)
{V$_{\mathrm{REF}}$~~(Reference Voltage Input)} ;
\node[right] at (IC.pin 16)
{V${}_{\mathrm{DD}}$~~(\SI{2.7}{\volt} --
\SI{5,5}{\volt} Power Supply)} ;
\end{circuitikz}   \end{document}
```

Kommunikation im World Wide Web

Abbildung 8.10 auf Seite 340 (HTTP Kommunikation zwischen client und server) kann mit folgendem LaTeX-Code erzeugt werden. Man braucht das Paket tikzpeople.

```
% Präambel
\documentclass[a4paper,12pt,parskip]{scrartcl}
\usepackage[ngerman]{babel} \usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}   \pagestyle{empty}
\usepackage{tikz, tikzpeople}
% document-Umgebung
\begin{document}   \begin{tikzpicture}
\node[draw] at (-4,3) {server 1};
\node[draw] at (0,3) {server 2};
\node[right,green!50!black] at (1.5,3) {resource (HTML-Datei)};
\draw[green!50!black,-Stealth] (1.6,3)--(1.1,3);
%\node[draw] at (4,3) {Server 3};
\node[draw] at (0,-0.4) {client};
\draw[purple,ultra thick] (0,-0.4) ellipse (12mm and 6mm);
\node[purple,left] at (-1.2,-0.4) {\small host = Rechner im Netz};
\node[purple,right] at (1.2,-0.4) {mein Rechner};
\node[alice,mirrored,shirt=red,undershirt=teal,minimum size=1.8cm]
at (-0.05,-2.4) {Ich};
\draw[-Stealth,teal!50!white,ultra thick] (-0.4,0.4)--(-0.4,2.4);
\draw[-Stealth,brown!60!white,ultra thick] (0.4,2.4)--(0.4,0.4);
\node[left] at (-0.4,1.4) {\textcolor{teal!70!white}{request}};
\node[right] at (0.4,1.4) {\textcolor{brown!80!white}{response}};
\node[align=left] at (-3.5,-2) {hier: client =\\web browser};
\end{tikzpicture}   \end{document}
```

9.2 Internet-Adressen

Kostenlose Literatur im Internet zum Raspberry Pi und seine Programmierung

Der Raspberry Pi Beginner's Guide von Gareth Halfacree stellt den Raspberry Pi vor. Das Buch kann (in Englisch) online gelesen oder heruntergeladen werden. Es wurde 2020 herausgegeben und die Hinweise zum Betriebssystem sind daher nicht mehr ganz aktuell.

Linux Journey ist ein einführender online Linuxkurs mit Übungen (Webseite in Englisch). William Shotts zeigt das Arbeiten mit der Kommandozeile in The Linux Command Line (PDF-Datei, in Englisch).

Das GNU Bash manual (in Englisch) kann man als Textdatei oder im PDF-Format herunterladen. Es kann auch gut als HTML-Webseite mit einem einfachen Browser wie `w3m` (siehe Seite 209) gelesen werden. Der Advanced Bash-Scripting Guide von Mendel Cooper vertieft das Arbeiten mit der `bash` (PDF-Datei, in Englisch). Nützlich ist auch die Webseite von Hund, A collection of handy ways of manipulating text in Bash.

Die Webseite https://docs.python.org/3/tutorial/ bietet ein online Python Tutorial (in Englisch) und auf der Webseite https://pythonbooks.org/free-books/ gibt es kostenlose Bücher über Python zum Herunterladen (in Englisch).

API (Programmierschnittstellen) ohne Anmeldung

JSON Daten zum „Astronomy Picture of the Day“, mit URL des Bilds
https://api.nasa.gov/planetary/apod?api_key=DEMO_KEY

JSON Daten zu Wechselkursen, zum Beispiel für den Euro:
https://cdn.jsdelivr.net/gh/fawazahmed0/currency-api@1/latest/currencies/eur.json

JSON Daten zu Wechselkursen, zum Beispiel für den US-Dollar:
https://open.er-api.com/v6/latest/USD

JSON Daten über die USA, zum Beispiel die Einwohnerzahl:
https://datausa.io/api/data?drilldowns=Nation&measures=Population

JSON Daten für einen zufällig erzeugten Nutzer, mit URL eines Photos
https://randomuser.me/api/

JSON Daten für den neuesten xkcd Webcomic, mit URL des Bilds
http://xkcd.com/info.0.json

DAWUM - Wahlumfragen in Deutschland, deutsch
https://api.dawum.de/

Witze in verschiedenen Sprachen, im JSON Format oder als einfacher Text, zum Beispiel für deutsche Witze als einfacher Text (englisch ist besser):
https://v2.jokeapi.dev/joke/Any?lang=de&format=txt

Web-Feeds

Ein Web-Feed (englisch: web feed) ist ein Dokument als Betriebsmittel eines Servers im Internet, welches aktuelle Informationen (insbesondere Nachrichten) in einem bestimmten Format (meistens Atom, JSON Feed oder RSS) bereitstellt. Sie können mit einem Feedreader (englisch: feed reader oder news aggregator) gelesen werden, zum Beispiel mit Newsboat, siehe Seite 215.

Hier einige Adressen (URL) für Web-Feeds:

AD HOC NEWS – Finanzzeitung:
https://www.ad-hoc-news.de/rss/nachrichten.xml

AD HOC NEWS – Wirtschaft:
https://www.ad-hoc-news.de/rss/wirtschaft.xml

AD HOC NEWS – Wissenschaft:
https://www.ad-hoc-news.de/rss/wissenschaft.xml

BBC News:
http://newsrss.bbc.co.uk/rss/newsonline_world_edition/front_page/rss.xml

UC Berkeley – physics news:
http://http://news.berkeley.edu/topics/physics/feed

China Daily – china:
http://www.chinadaily.com.cn/rss/china_rss.xml

China Daily – world:
http://www.chinadaily.com.cn/rss/world_rss.xml

Defence Blog (US-based):
http://www.defence-blog.com/feed

Deutsche Bundesregierung:
https://www.bundesregierung.de/service/rss/breg-de/1151244/feed.xml

Deutscher Bundestag:
https://www.bundestag.de/static/appdata/includes/rss/pressemitteilungen.rss

CBC (Canadian Broadcasting Corporation):
https://rss.cbc.ca/lineup/topstories.xml

Der Spiegel – Schlagzeilen:
https://www.spiegel.de/schlagzeilen/tops/index.rss

Deutsche Welle (DW):
https://rss.dw.com/atom/rss-de-all

Web-Feeds der DW zu besonderen Themen findet man auf der Webseite https://www.dw.com/de/rss/s-9773

Fox News – world:
https://moxie.foxnews.com/feedburner/world.xml

https://www.huffpost.com/ (US-amerikanische Onlinezeitung):
https://www.huffpost.com/section/front-page/feed

Jenaer Nahverkehr:
https://www.nahverkehr-jena.de/index.php?id=37&type=100

Jenapolis:
https://coolis.de/category/news-im-osten/thueringen/jenapolis/rss

Ostthüringer Zeitung – Jena:
https://www.otz.de/regionen/jena/rss

Physics World:
https://physicsworld.com/feed

Rheinische Post:
https://rp-online.de/feed.rss

ScienceDaily Top Science News:
https://www.sciencedaily.com/rss/all.xml

ScienceDaily Computers & Math:
https://www.sciencedaily.com/rss/computers_math.xml

ScienceDaily Health & Medicine:
https://www.sciencedaily.com/rss/health_medicine.xml

ScienceDaily Space & Time:
https://www.sciencedaily.com/rss/space_time.xml
Schweizer Radio und Fernsehen (SRF) – internationale Nachrichten:
http://www.srf.ch/news/bnf/rss/1922
tagesschau.de:
https://www.tagesschau.de/xml/atom/
TASS russian news agency:
https://tass.com/rss/v2.xml
The Guardian:
https://www.theguardian.com/rss
Thüringer Allgemeine:
https://www.thueringer-allgemeine.de/rss
Tichys Einblick:
https://www.tichyseinblick.de/rss
Washington Times – political analysis:
https://www.washingtontimes.com/atom/headlines/news/analysis/
Washington Times – world news:
https://www.washingtontimes.com/atom/headlines/news/world/

Webseiten für textbasierte Webbrowser

AHG: Linux auf dem Raspberry Pi (deutsch): https://ahg88.webnode.page/

Canadian Broadcasting Corp. Nachrichten (englisch): https://www.cbc.ca/lite/news

CNN Nachrichten (englisch): https://lite.cnn.com/en

DuckDuckGo Suchmaschine (englisch): https://duckduckgo.com/lite/

EIKE Blog der „Klima-Gegner“ (deutsch): https://eike-klima-energie.eu/

engineering blogs – for computer freaks (englisch): https://engineeringblogs.xyz/

GNU/Linux (englisch, teilweise deutsch): http://www.gnu.org/

Google Suchmaschine (deutsch): https://www.google.de/

Legible News (englisch, mit Verweisen auf Wikipedia): https://legiblenews.com/

Literature – free ebooks (englisch): http://gutenberg.net.au/index.html

Low-tech Magazine (englisch): https://solar.lowtechmagazine.com/

MetaFilter general-interest community blog (englisch): https://www.metafilter.com/

National Public Radio Nachrichten und Kultur (englisch): https://text.npr.org/

New York Times (englisch): https://www.nytimes.com

Python 3 Tutorial (deutsch): https://www.python-kurs.eu/python3_kurs.php

Routenplaner Auto, Bahn, Bus – für alle Länder (englisch): https://gdir.telae.net/

ScienceDaily – research news (englisch): https://www.sciencedaily.com/

Science Hobbyist – electrifying blog (englisch): http://amasci.com/

Süddeutsche Zeitung (deutsch): https://www.sueddeutsche.de/

Teletext-Nachrichten der ARD (deutsch): https://www.ard-text.de/mobil/100

Teletext-Nachrichten des MDR (deutsch): https://www.mdr.de/CONT/mobil/110.html

Wetterbericht (englisch): https://wttr.in/

Wiby – search for static, single-person-curated sites (englisch): https://wiby.me/

Wikipedia (deutsch): https://www.wikipedia.de/

68k.news – basic HTML Google News (englisch oder deutsch): http://68k.news/

Internet-Radiosender

Hinweise zu Adressen (URL) von Internet-Radiosendern findet man auf den Webseiten
https://streamurl.link/ (Radio stream URL search engine, liefert Stream URL)
https://www.radio-browser.info/ (RadioBrowser)
https://linupedia.org/opensuse/Radiosender (deutsche Sender, Linupedia.org)
https://linupedia.org/opensuse/Radiosender-1 (internationale, Linupedia.org)
https://radio-locator.com/ (USA und internationale, Radio-Locator.com.org)
https://wiki.ubuntuusers.de/Internetradio/Stationen/ (ubuntuusers.de)
https://wiki.ubuntuusers.de/Internetradio/Internetradio-Stationen/
http://dir.xiph.org/ (Icecast, im GUI-Browser: Play, Audio-Adresse kopieren)
https://vtuner.com/setupapp/guide/asp/BrowseStations/startpage.asp

Auf die oben genannte Sendersammlung RadioBrowser kann man mit dem Pythonmodul pyradios zugreifen. Außerdem kann man die Senderdaten in einer API (Programmierschnittstelle) im Format JSON oder XML oder als M3U playlist herunterladen, siehe
http://all.api.radio-browser.info/

Amazing Tales, Old Time Radio, USA, 32 kbit/s (OK 23.08.2023)
`https://onlineradiobox.com/json/us/amazingtales/play`

Audio Noir (USA), Drama (englisch), 48 kbit/s (OK 15.08.2022)
`http://198.245.61.123:8000/noir/`

Bayerischer Rundfunk (BR) Radio-Livestreams siehe:
https://www.br.de/service/urls-livestreams-100.html

Bayern 2 Nord, MP3 128 kbit/s (OK 22.04.2022)
`https://dispatcher.rndfnk.com/br/br2/nord/mp3/mid`
Kultursender des Bayerischen Rundfunks (BR) mit Musik und Information, Hörspiele

BR-Klassik, MP3 128 kbit/s (OK 22.04.2022)
`https://dispatcher.rndfnk.com/br/brklassik/live/mp3/mid`
Sender mit klassischer Musik des Bayerischen Rundfunks (BR)

BR24, MP3 128 kbit/s (OK 22.04.2022)
`https://dispatcher.rndfnk.com/br/br24/live/mp3/mid`
Nachrichtensender des Bayerischen Rundfunks (BR)

BR24live, MP3 128 kbit/s (OK 22.04.2022)
`https://dispatcher.rndfnk.com/br/br24live/live/mp3/mid`
Sender des Bayerischen Rundfunks (BR), überträgt große Sportereignisse (Fussball) und Sitzungen von Bundestag und Bayerischem Landtag, ansonsten das Programm von B24

BBC Radio 4, englisch, MP3 128 kBit/s, (OK 14.08.2022)
`http://stream.live.vc.bbcmedia.co.uk/bbc_radio_fourfm`

BBC World Service, news,talk, englisch, MP3 56 kBit/s (OK 14.08.2022)
`http://stream.live.vc.bbcmedia.co.uk/bbc_world_service`

Boom Radio, oldies, 128 kBit/s, OK 08.05.2022)
`https://listen-boomradio.sharp-stream.com/65_boom_radio_live_128`

British Comedy Radio, siehe `https://britishcomedyradio.org/how-to-tune-in/`

Capital FM 105.3 (Moskau), pop music, englisch, MP3 128 kBit/s (OK 14.08.2022)
`http://icecast.vgtrk.cdnvideo.ru/capitalfmmp3`

CBC Radio One, news and information, Canada (PLAYLIST, OK 15.08.2022)
`http://www.surfmusic.de/m3u/cbcv-cbc-radio-one-90-5-fm,15345.m3u`

CJNU 93.7FM - Nostalgia Radio (Canada), Oldies (englisch), 128 kbit/s (OK 15.08.2022)
`http://noasrv.caster.fm:10188/stream.m3u`

Classic Vinyl, 1930, 1940, 1950,1960ies beautiful music, englisch (OK 15.12.2023)
`https://icecast.walmradio.com:8443/classic`

Crime & Suspense, ROKiT Radio Network (OK 15.08.2022)
`http://streaming01.zfast.co.uk:8168/`

Deutschlandfunk (Dlf) Radio-Livestreams siehe:
`https://www.deutschlandfunk.de/unsere-live-streams.2396.de.html`

Deutschlandfunk (OK 22.10.2021)
`https://st01.sslstream.dlf.de/dlf/01/128/mp3/stream.mp3?aggregator=web`

FOX News, englisch, MP3 128 kBit/s (OK 15.08.2022)
`http://tunein.streamguys1.com/foxnews`

Hessischer Rundfunk hr 1 Livestreams siehe:
`https://www.hr1.de/service/livestream,livestream-114.html`
Pop- und Rockmusik der 1980er, Reportagen aus Hessen und der Welt

Hessischer Rundfunk Info-Radio Livestreams siehe:
`https://www.hr-inforadio.de/service/livestream,hr-info-livestream-100.html`

https://laut.fm/hoerbuchzauber, deutsch (OK 14.08.2022)
`http://stream.laut.fm/hoerbuchzauber`

Hörspielprojekt, deutsch (OK 15.08.2022)
`http://stream.laut.fm:80/hoerspiel`

Infowars, conspiracy talk, englisch (OK 14.08.2022)
`http://streams1.infowars.com/stream/2/`

Kontrafunk, liberal-konservative Nachrichten und Talk, deutsch (OK 15.12.2023)
`https://s5.radio.co/sca4082ebb/listen`

LBC UK, talk, englisch, AAC 48 kbit/s (OK 15.08.2022)
`http://media-ice.musicradio.com/LBCUK`

MDR KLASSIK Livestreams siehe:
`https://www.mdr.de/mdr-klassik-radio/livestream-mdr-klassik--100.html`

MDR KULTUR Livestreams siehe:
`https://www.mdr.de/kultur/radio/livestream-mdr-kultur-100.html`

Mystery Play Internet Radio, englisch (OK 15.08.2022)
`http://198.178.123.11:7080/`

NBC News Radio (PLAYLIST, OK 15.08.2022)
`http://www.surfmusic.de/m3u/24-7-news,18547.m3u`

NDR Radio-Livestreams siehe:
`https://www.ndr.de/service/Die-Radio-Livestream-Links,livestreams101.html`

NDR Kultur (OK 24.09.2021)
`http://www.ndr.de/resources/metadaten/audio/m3u/ndrkultur.m3u`

Nigeria Info, talk, music, englisch, MP3 128 kBit/s (OK 14.08.2022)
`https://stream.coolwazobiainfo.com/niginf_lag`

NPR 24 Hour Program (OK 13.05.2022)
`https://npr-ice.streamguys1.com/live.mp3`

NPR Hourly News Summary, Washington DC (OK 13.05.2022)
`http://public.npr.org/anon.npr-mp3/npr/news/newscast.mp3`

NSF Science360 Radio, englisch (OK 15.08.2022)
`http://amber.streamguys1.com:4200/live`

Radio Bremen Radio-Livestreams siehe:
`https://www.radiobremen.de/service/hilfe/livestreams-100.html`

Radio Bremen 1 (OK 05.11.2021)
`http://icecast.radiobremen.de/rb/bremeneins/live/mp3/64/stream.mp3`

Radio Caroline, 52 kBit/s (PLAYLIST, OK 15.08.2022)
`http://sc2.radiocaroline.net:8010/listen.pls`

Radio LOTTE (Weimar) (OK 14.08.2020)
`http://www.radiolotte.de/stream/radiolotte.m3u`

Radio Nordseewelle Radio-Livestreams siehe:
`http://stream.radio-nordseewelle.de/`

Radio OKJ - Bürgerradio für Jena (OK 14.01.2022)
`https://radio-okj.de/wp-content/themes/radiookj/library/livestream_radio_okj.m3u`

Radio SRF 2 Kultur (OK 06.08.2020)
`http://stream.srg-ssr.ch/m/drs2/mp3_128`

Radio SRF 4 News (OK 06.08.2020)
`http://stream.srg-ssr.ch/m/drs4news/mp3_128`

Radio Waltl, freies Talk-Radio (OK 15.08.2022)
`http://live.datamatix.at:8024`
„das kleinste Kulturradio der Welt“

rbb Kultur (OK 14.01.2022)
`http://rbbkultur.de/livemp3`
URL-Info: `https://www.rbb-online.de/rbbkultur/hilfe/kulturradio-online-empfangen.html`

Relic Radio On The Air, drama from the late 1930's to the early 1970's (OK 5.08.2022)
`http://s8.voscast.com:7372`

Rick Future FM, Science Fiction-Hörspielserie, deutsch (INAKTIV)

Science Fiction & Supernatural, ROKiT Classic Radio, englisch (OK 15.08.2022)
`http://streaming02.zfast.co.uk:8110/`

Sky News Radio Live (Australia), news, talk, englisch, AAC+ 49 kBit/s (OK 14.08.2022)
`https://ais-arn.streamguys1.com/au_033/playlist.m3u8`

SKY NEWS Hourly News Brief, London (OK 13.05.2022)
`http://video.news.sky.com/snr/news/snrnews.mp3`

Soundtalesproductions, Hörspielsender, deutsch (OK 14.08.2022)
`http://stream.laut.fm/soundtalesproductions`

Vermont Public Radio (OK 15.08.2022)
`http://vpr.streamguys.net/vpr96.mp3`

Vorleser.net Radio-Livestreams siehe:
`https://www.vorleser.net/projekte-a-814.html`

Westdeutscher Rundfunk (WDR) Radio-Livestreams siehe:
`https://www1.wdr.de/unternehmen/der-wdr/empfang-technik/webradio-100.html`

WDR 5, news talk, deutsch, 128 kBit/s (OK 14.08.2022)
`http://wdr-wdr5-live.icecast.wdr.de/wdr/wdr5/live/mp3/128/stream.mp3`

WYSL 1040 AM talk radio, englisch (OK 15.08.2022)
`http://64.78.234.165:8000/WYSL`

WSKG, englisch, MP3-Stream (OK 15.08.2022)
`https://wskg.streamguys1.com/wskg`

WTAQ in Wisconsin, „News, Talk, Sports“, 32 kbit/s MP3-Stream (OK 15.12.2023)
`https://playerservices.streamtheworld.com/api/livestream-redirect/WTAQAM.mp3`

1920s Radio Network (OK 15.08.2022)
http://kara.fast-serv.com:8358/

1940s Radio, ROKiT Radio Network (OK 15.08.2022)
`http://streaming03.zfast.co.uk:8256/`

80s80s Radio-Livestreams siehe:
`http://streams.80s80s.de/`

Webcams an öffentlichen Orten

Wetterstation Beutenberg-Campus, Jena (wechselnde Bilder im Grafikformat JPEG)
`https://www.bgc-jena.mpg.de/wetter/towercam.jpg`

Domplatz, Erfurt (wechselnde Einzelbilder im Grafikformat JPEG)
`https://www.erfurt.de/webcam/domplatz.jpg`

Fischmarkt, Erfurt (wechselnde Einzelbilder im Grafikformat JPEG)
`https://www.erfurt.de/webcam/fischmarkt.jpg`

Webcam am Roten Rathaus, Berlin (wechselnde Einzelbilder im Grafikformat JPEG)
`https://www.berlin.de/webcams/rathaus/webcam.jpg`

Skyline von San Francisco mit Financial District, San Francisco
(wechselnde Einzelbilder im Grafikformat JPEG)
`https://castrocam.net/img/castrocamHD.jpg`

San Francisco, Blick auf Castro Street (Livestream, Cam 1), YouTube
`https://www.youtube.com/watch?v=JjbMA7dSN2A`

San Francisco, Blick auf Castro Street (Livestream, Cam 2), YouTube
`https://www.youtube.com/watch?v=tik0iS8ftuc`

San Francisco Skyline (Livestream von Webcam auf Treasure Island), YouTube
`https://www.youtube.com/watch?v=K3vjVPiXq5g`

New York City, Times Square (Livestream), YouTube
`https://www.youtube.com/watch?v=1-iS7LArMPA`

Eine Pseudo-Webcam (etwa einstündiges Video ohne Ton):
Rio de Janeiro, Copacabana Beach (Brasilien)
`https://video2archives.earthcam.com/archives/_definst_/MP4:permanent/6593/2017/10/01/1300.mp4/playlist.m3u8`

Der Deutsche Wetterdienst (DWD) betreibt einige Webcams, die schöne, regelmäßig aktualisierte Bilder in verschiedenen Formaten bereitstellen, siehe
`https://opendata.dwd.de/weather/webcam/`

Internet-Fernsehsender

Umfassende, aktualisierte Listen mit Internetfernsehsendern gibt es auf der Webseite
https://github.com/iptv-org/iptv Hier einige Beispiele:

Cheddar News, englisch (OK 28.06.2022)
`https://live.chdrstatic.com/cbn/primary/1.m3u8`

Classic Cinema, englisch, Auflösung 240p (OK 28.06.2022)
`https://rpn1.bozztv.com/36bay2/gusa-classiccinema/mono.m3u8`

Radio Bremen Fernsehen, deutsch (OK 28.06.2022)
`https://rbhlslive.akamaized.net/hls/live/2020435/rbfs/master.m3u8`

ZDF neo, deutsch, Auflösung 1280 x 720 (OK 28.06.2022)
`https://zdf-hls-16.akamaized.net/hls/live/2016499/de/high/master.m3u8`

Mediatheken

Die folgende Liste ist eine Sammlung von Webseiten, die Bücher, Hörbücher, Bilder, Videofilme oder ähnliches anbieten, davon vieles kostenlos. Ob die Urheberrechte gewahrt werden, habe ich nicht untersucht und jeder Nutzer hat dies daher selbst zu prüfen.

ARD Mediathek, siehe: https://www.ardmediathek.de/

ARD-alpha Mediathek, siehe: https://www.sendungverpasst.de/ard-alpha

ARD ONE Mediathek, siehe: https://www.ardmediathek.de/one/

ARTE Mediathek siehe: https://www.arte.tv/de/

Australian Radio Serials and Sponsors Programs, siehe: http://www.crossbandradio.com/ozserials
old time radio

Bayerischer Rundfunk (BR) Mediathek, siehe: https://www.br.de/mediathek/

BookRix, siehe: https://www.bookrix.com/books.html
englischsprachige Bücher

Buten un binnen Mediathek, siehe: https://www.butenunbinnen.de/livestream/
Lokalnachrichten (TV) aus Bremen

Detektei Suni & Partner – kostenlose Hörspielserie für Kinder und Jugendliche, URL:
https://schulze-froehlich.com/projekte/detektei-suni-partner-die-kostenlose-hoerspielserie/
(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Engineering Books PDF, siehe: https://engineeringbookspdf.com/ free download of books (PDF) in science and engineering – registration not necessary

Internet Archive, siehe: https://archive.org/
Digitale Bibliothek von digitalisierten Büchern, Audiodateien, Filmen und mehr

Jamendo Music, siehe: https://www.jamendo.com/explore

Kjos Music, siehe: https://kjos.com/multimedia-library/

LibriVox, siehe: https://librivox.org/
free public domain audiobooks, read by volunteers from around the world

ManyBooks, siehe: https://manybooks.net/
free books for online reading or – after registration – download in various formats

NASA Mediathek, siehe: https://www.dvidshub.net/unit/NASA

Norddeutscher Rundf. (NDR) Mediathek, siehe: https://www.ardmediathek.de/ndr/

NSF Mediathek, siehe: https://www.nsf.gov/news/mmg/index.jsp
US National Science Foundation Wissenschaftsmedien, nutze `Advanced Search`

n-tv Mediathek, siehe: https://www.n-tv.de/mediathek/videos/
Nachrichten-Videos

OTR.Network Library (BETA), siehe: http://www.otr.net/
old time radio, shows and drama series

Project Gutenberg, siehe: https://www.gutenberg.org/
kostenlose digitale Bibliothek zum Herunterladen von E-Books in mehreren Formaten

Projekt Gutenberg-DE, siehe: https://www.projekt-gutenberg.org/
kostenlose digitale Bibliothek zum onlne Lesen deutschsprachiger alter Bücher

Vorleser Mediathek, siehe: https://www.vorleser.net/
kostenlose deutsche MP3-Hörbücher und Hörspiele

Rennsteig TV Mediathek, siehe: https://www.rennsteig.tv/auf-sendung/
Lokalnachrichten (TV) vom Rennsteig (Thüringen)

ZDF Mediathek, siehe: https://www.zdf.de/

Z-library, Englisch: https://z-lib.org/, Deutsch: https://de.z-lib.org/
free download of books in various languages (auch Deutsch) – registration not necessary

E-Books in der Mediathek Project Gutenberg

Victor Appleton (1961)):
Tom Swift and the Visitor from Planet X,
EPUB ohne Bilder: https://www.gutenberg.org/ebooks/17985.epub.noimages

Lewis Carroll (1865):
Alice's Adventures in Wonderland,
EPUB ohne Bilder: https://www.gutenberg.org/ebooks/11.epub.noimages

Agatha Christie (1923):
The Plymouth Express Affair,
EPUB ohne Bilder: https://www.gutenberg.org/ebooks/66446.epub.noimages

Agatha Christie (1925):
The Secret of Chimneys,
EPUB ohne Bilder: https://www.gutenberg.org/ebooks/65238.epub.noimages

Benjamin Harrow (1920):
From Newton to Einstein: Changing Conceptions of the Universe,
EPUB ohne Bilder: https://www.gutenberg.org/ebooks/60271.epub.noimages

Max Planck (1922):
Vorlesungen über Thermodynamik,
PDF: https://gutenberg.org/files/31564/31564-pdf.pdf

Max Simon (1909):
Geschichte der Mathematik im Altertum in Verbindung mit antiker Kulturgeschichte,
EPUB mit Bildern: https://www.gutenberg.org/ebooks/62131.epub.images

Jonathan Swift (1704):
The Battle of the Books, and other Short Pieces,
EPUB ohne Bildern: https://www.gutenberg.org/ebooks/623.epub.noimages

Bücher in der Mediathek Internet Archive

Greg Green: Cannabis Grow Bible, 4th Edition (2001), PDF, 8,9 MB, URL:
https://archive.org/download/CannabisGrowBible4thEdition/Cannabis%20Grow%20Bible%2C%204th%20Edition.pdf

B. Van Brunt: The Calculus of Variations (2004), PDF, 2,3 MB, URL:
https://archive.org/download/springer_10.1007-b97436/10.1007-b97436.pdf

Universe – Britannica Illustrated Science Library (2008), PDF, 48,4 MB, URL:
https://archive.org/download/universe-britannica-illustrated/Universe%20%28Britannica%20Illustrated%20%29.pdf

CRC Handbook of Chemistry and Physics (2016), PDF, 62,7 MB, URL:
https://archive.org/download/w.-m.-haynes-al.-2016-crc-handbook-of-chemistry-and-physics/W.M.%20Haynes%20%26%20al.%202016%20-%20CRC%20Handbook%20of%20chemistry%20and%20physics.pdf

Audiodateien in der Mediathek Internet Archive

Man beachte: In der folgenden Liste wurden URL umgebrochen, damit sie auf die Druckbreite passen.

Ibrahim Efendi: Tchik dérim djana tchikmaz ténimden (1909), 5,5 MB, URL:
https://archive.org/download/78_none-legible_gbia0182005b/None%20legible.mp3

Enrico Caruso, Galli-Curci, Flora Perini, Giuseppe De Luca: Bella figlia dell'amore (1917), 3,9 MB, URL:
https://archive.org/download/CarusoAmelitaGalli-CurciFloraPeriniGiuseppedeLuca/CarusoAmelitaGalli-CurciFloraPeriniGiuseppedeLuca-BellaFigliaDellamore.mp3

Whispering Jack Smith: All By Yourself In The Moonlight (1928), 7,0 MB, URL:
https://archive.org/download/WhisperingJackSmithCollection1925-1932/AllByYourselfInTheMoonlight1928WhisperingJackSmithWithCarrollGibbonsOrchestra.mp3

Annette Hanshaw: Get Out and Get Under the Moon (1928), 2,6 MB, URL:
https://archive.org/download/yt-5s.com-get-out-and-get-under-the-moon-remastered-128-kbps/yt5s.com%20-%20Get%20out%20and%20Get%20Under%20the%20Moon%20%28Remastered%29%20%28128%20kbps%29.mp3

Mae Questel: Don't Take My Boop-Oop-A-Doop Away (1933), 12,2 MB, URL:

https://archive.org/download/dont-take-my-boop-boop-be-doop-away-by-mae-questel-boop-boop-be-doop/Don%27t%20Take%20My%20Boop%20Boop%20Be%20Doop%20Away%20by%20Mae%20Questel%20%28Boop-Boop-Be-Doop%29.mp3

Riley Puckett: Isle of Capri (1935), 3,4 MB, URL:
https://archive.org/download/the-isle-of-capri/The%20Isle%20Of%20Capri%20%28remaster%202%29.mp3

Mae Questel: I Want You For Christmas (1937), 6,4 MB, URL:
https://archive.org/download/mae-questel-the-betty-boop-girl-i-want-you-for-christmas/Mae%20Questel%20%28The%20Betty%20Boop%20Girl%29%20-%20I%20Want%20You%20For%20Christmas.mp3

Charles Trenet: Fleur bleue (1937), 2,1 MB, URL:
https://archive.org/download/78_fleur-bleue-blue-flower_charles-trenet-wal-berg_gbia0012131b/Fleur%20Bleue%20%28Blue%20Flower%29%20-%20Charles%20Trenet-restored.mp3

Judy Garland: Over the Rainbow (1939), 2,3 MB, URL:
https://archive.org/download/78_over-the-rainbow_judy-garland-victor-young-and-his-orchestra-harold-arlen-e.-y.-har_gbia0004863a/Over%20The%20Rainbow%20-%20Judy%20Garland%20-%20Victor%20Young%20and%20his%20Orchestra-restored.mp3

Golden Gate Quartet: Didn't It Rain (1941), 2,6 MB, URL:
https://archive.org/download/78_didnt-it-rain_golden-gate-quartet_gbia0010231a/Didn%27t%20It%20Rain%20-%20Golden%20Gate%20Quartet-restored.mp3

The Andrews Sisters: Boogie Woogie Bugle Boy (1941), 2,5 MB, URL:
https://archive.org/download/78_boogie-woogie-bugle-boy_andrews-sisters-with-vic-schoen-and-his-orchestra-don-raye_gbia0033285a/Boogie%20Woogie%20Bugle%20Boy%20-%20Andrews%20Sisters%20With%20Vic%20Schoen%20And%20His%20Orchestra-restored.mp3

Bing Crosby: Remember Hawaii (1942), 4,7 MB, URL:
https://archive.org/download/78_remember-hawaii_bing-crosby-dick-mcintire-and-his-harmony-hawaiians-meredith-willso_gbia0010923b/Remember%20Hawaii%20-%20Bing%20Crosby%20-%20Dick%20McIntire%20and%20His%20Harmony%20Hawaiians.mp3

The Andrews Sisters: Strip Polka (1942), 5,8 MB, URL:
https://archive.org/download/78_strip-polka_the-andrews-sisters-vic-schoen-his-orchestra-mercer_gbia7023373b/STRIP%20POLKA%20-%20THE%20ANDREWS%20SISTERS%20-%20Vic%20Schoen%20%26%20his%20Orchestra.mp3

The Andrews Sisters: Mr. Five by Five (1942), 3,7 MB, URL:
https://archive.org/download/78_mister-five-by-five_andrews-sisters-vic-schoen-and-his-orchestra-gene-de-paul-don-r_gbia0011930b/Mister%20Five%20By%20Five%20-%20Andrews%20Sisters%20-%20Vic%20Schoen%20and%20His%20Orchestra.mp3

The Andrews Sisters: Rum and Coca-Cola (1944), 4,1 MB, URL:
https://archive.org/download/rum-and-coca-cola/Rum%20And%20Coca-Cola%20%28remix%202%29.mp3

Bobby Capó: Julian el Zapatero (1945), 2,8 MB, URL:
https://archive.org/download/78_julian-el-zapatero-julian-the-shoemaker_bobby-ca

po-la-orquesta-internacional-seec_gbia0020604a/Julian%20el%20Zapatero%20%28Julian%20The%20Shoemaker%29%20-%20Bobby%20Capo-restored.mp3

The Andrews Sisters: The Bride and Groom Polka (1947), 4,2 MB, URL:
https://archive.org/download/78_the-bride-and-groom-polka_the-andrews-sisters-vic-shoen-his-orchestra-lee-roberts_gbia3005330b/THE%20BRIDE%20AND%20GROOM%20POLKA%20-%20THE%20ANDREWS%20SISTERS.mp3

Janette Davis: They Can't Make a Lady Out of Me (1947), 2,7 MB, URL:
https://archive.org/download/78_they-cant-make-a-lady-out-of-me_janette-davis-jaffe-tobias-archie-bleyer_gbia0018118a/They%20Can%27t%20Make%20a%20Lady%20Out%20of%20Me%20-%20Janette%20Davis-restored.mp3

Two Ton Baker: I'm a Lonely Little Petunia In an onion patch (1947), 3,7 MB, URL:
https://archive.org/download/78_im-a-lonely-little-petunia-in-an-onion-patch_dick-two-ton-baker-and-his-music-ma_gbia0070068b/I%27m%20a%20Lonely%20L%20-%20Dick%20%22Two%20Ton%22%20Baker%20And%20His%20Music%20Makers-restored.mp3

Charles Trenet: Formidable (1949), 2,3 MB, URL:
https://archive.org/download/78_formidable_charles-trenet-lasry-albert-lasry_gbia0011637a/Formidable%20-%20Charles%20Trenet%20-%20Lasry%20-%20Albert%20Lasry-restored.mp3

Édith Piaf: La vie en rose (1950), 3,5 MB, URL:
https://archive.org/download/78_la-vie-en-rose_edith-piaf-louiguy-luypaerts_gbia0011638b/La%20Vie%20en%20Rose%20-%20Edith%20Piaf%20-%20Louiguy%20-%20Luypaerts-restored.mp3

Georgia Gibbs: Kiss of Fire (1952), 2,4 MB, URL:
https://archive.org/download/78_kiss-of-fire_georgia-gibbs-glenn-osser-lester-allen-robert-hill_gbia0012793b/Kiss%20of%20Fire%20-%20Georgia%20Gibbs%20-%20Glenn%20Osser-restored.mp3

Georgia Gibbs: Seven Lonely Days (1953), 2,2 MB, URL:
https://archive.org/download/78_seven-lonely-days_georgia-gibbs-glenn-osser-schuman-brown-georgia-gibbs-and-the-yal_gbia0004941b/Seven%20Lonely%20Days%20-%20Georgia%20Gibbs%20-%20Glenn%20Osser-restored.mp3

Eartha Kitt: Uska Dara (A Turkish Tale) (1953), 5,0 MB, URL:
https://archive.org/download/78_uska-dara-a-turkish-tale_eartha-kitt-henri-ren-and-his-orchestra_gbia0262228b/USKA%20DARA%20-%20A%20TURKISH%20TALE%20-%20Eartha%20Kitt.mp3

Eartha Kitt: Under the Bridges of Paris (1954), 2,6 MB, URL:
https://archive.org/download/78_under-the-bridges-of-paris_cochran-scotto-rodor-eartha-kitt-with-henri-rene-and-his_gbia0001251b/Under%20The%20Bridges%20of%20Paris%20-%20Cochran%20-%20Scotto-restored.mp3

Eartha Kitt: The Heel (1955), 4,1 MB, URL:
https://archive.org/download/78_the-heel_eartha-kitt-henri-rene-and-his-orchestra-willard-robison-lee-wilson-leo-fe_gbia0103209a/The%20Heel%20-%20Eartha%20Kitt%20-%20Henri%20Rene%20and%20his%20Orchestra.mp3

Nat Charles and Orchestra: The Crazy Otto Medley (1955), 5,2 MB, URL:
https://archive.org/download/78_the-crazy-otto-how-important-can-it-be_nat-charles-mimi-martel-the-four-rhythmaire_gbia0016335b/The%20Crazy%20Otto%3B%20How%20Important%20Can%20It%20Be%20-%20Nat%20Charles-restored.mp3

Somethin' Smith and the Redheads: Pretty Baby (1955), 2,3 MB, URL:
https://archive.org/download/78_pretty-baby_somethin-smith-the-redheads-kahn-t-jackson-van-alstyne_gbia0052860a/Pretty%20Baby%20-%20Somethin%20Smith%20%26%20the%20Redheads-restored.mp3

Dean Martin: Memories Are Made of This (1955), 2,1 MB, URL:
https://archive.org/download/78_memories-are-made-of-this_dean-martin-dick-stabile-gilkyson-dehr-miller_gbia0012329a/Memories%20Are%20Made%20of%20This%20-%20Dean%20Martin%20-%20Dick%20Stabile-restored.mp3

Elvis Presley: Don't Be Cruel (1956), 2,2 MB, URL:
https://archive.org/download/78_dont-be-cruel_elvis-presley-otis-blackwell_gbia0079252b/Don%27t%20Be%20Cruel%20-%20Elvis%20Presley%20-%20Otis%20Blackwell-restored.mp3

Teresa Brewer: Crazy with Love (1956), 3,1 MB, URL:
https://archive.org/download/78_crazy-with-love_teresa-brewer-aaron-schroeder-josephine-peoples-dick-jacobs_gbia0160657b/CRAZY%20WITH%20LOVE%20-%20TERESA%20BREWER%20-%20Aaron%20Schroeder.mp3

Jerry Lee Lewis: Great Balls of Fire (1957), 2,1 MB, URL:
https://archive.org/download/78_great-balls-of-fire_jerry-lee-lewis-and-his-pumping-piano-hammer-blackwell_gbia0060105b/Great%20Balls%20of%20Fire%20-%20Jerry%20Lee%20Lewis%20And%20His%20Pumping%20Piano-restored.mp3

The Chordettes: Come Home To My Arms (1957), URL:
https://archive.org/download/78_come-home-to-my-arms_the-chordettes-e-jane-l-bagu ley-archie-bleyer_gbia0322063a/COME%20HOME%20TO%20MY%20ARMS%20-%20THE%20CHORDETTES%20-%20E.%20Jane.mp3

Little Richard: Good Golly, Miss Molly (1958), 2,6 MB, URL:
https://archive.org/download/78_good-golly-miss-molly_little-richard-marascalco-blackwell_gbia0060544a/Good%20Golly%20Miss%20Molly%20-%20Little%20Richard%20-%20Marascalco-restored.mp3

The Chordettes: Lollipop (1958), 3,2 MB, URL:
https://archive.org/download/78_lollipop_the-chordettes-j-dixon-b-ross-archie-bleyer_gbia0068558a/Lollipop%20-%20The%20Chordettes%20-%20J.%20Dixon%20-%20B.%20Ross.mp3

Dean Martin: Let It Snow! (1959), 4,6 MB, URL:
https://archive.org/download/dean-martin-let-it-snow-let-it-snow-let-it-snow/Dean%20Martin%20-%20Let%20It%20Snow%21%20Let%20It%20Snow%21%20Let%20It%20Snow%21.mp3

Édith Piaf: Non, je ne regrette rien (1960), 3,1 MB, URL:
https://archive.org/download/01-edith-piaf-columbia-c-21725-je-ne-regrette-rien/01%20-%20Edith%20Piaf%20Columbia%20C%2021725%20-%20Je%20ne%20regrette%20rien.mp3

María de Jesús Vásquez Vásquez: Que Chiquitito Esta Mi Corazon (1961), 3,0 MB, URL:
https://archive.org/download/78_que-chiquitito-esta-mi-corazon_ma-de-jesus-vasquez-trio-los-soberanos-gustavo-torr_gbia0020531b/Que%20Chiquitito%20Esta%20Mi%20Corazon%20-%20Ma.%20de%20Jesus%20Vasquez.mp3

Freddy Quinn: Junge, komm bald wieder (1962), 4,1 MB, URL:
https://archive.org/download/freddy-polydor-24981/01%20FREDDY%20-%20Polydor%2024981%20A%20-%20Junge%2C%20komm%20bald%20wieder%20

France Gall: Poupée de cire, poupée de son (1965), 3,4 MB, URL:
https://archive.org/download/france-gall-philips-437.032-be/01%20-%20France%20GALL%20-%20Philips%20437.032%20BE%20-%20Side%20A%20-%20Poup%C3%A9e%20de%20Cire%2C%20Poup%C3%A9e%20de%20Son%20%28Gainsbourg%29.mp3

Simon & Garfunkel: Homeward Bound (1966), 5,8 MB, URL:
https://archive.org/download/01-scarborough-fair-canticle_202011/04-Homeward%20Bound.mp3

Nancy Sinatra: You Only Live Twice (1967), 3,7 MB, URL:
https://archive.org/download/nancy_sinatra-you-only-live-twice_single/A1-You%20Only%20Live

Flamin' Groovies: Babes In The Sky (1968), 2,6 MB, URL:
https://archive.org/download/the-flamin-groovies-supersneakers-1968/02%20-%20Babes%20In%20The%20Sky.mp3

Dimension X - single episodes (Veröffentlichung: 1950 – 1951), 50 Audiodateien, URL:
https://archive.org/details/OTRR_Dimension_X_Singles

Science Fiction Radio; Atom Age Adventures (1938 – 1962), 52 Audiodateien, URL:
https://archive.org/details/Science_Fiction_Radio_Atom_Age_Adventures

Filme in der Mediathek Internet Archive

Filme für Kino und Fernsehen aus der guten alten Zeit findet man im Internet Archive. Voraussetzung ist eine Farbmonitor, der auch Schwarz-Weiß-Filme wiedergeben kann ;-)

Cruel, Cruel Love (comedy silent film, 1914), 46,4 MB, URL:
https://archive.org/download/silent-cruel-cruel-love/Cruel%2C%20Cruel%20Love.mp4

The Golf Specialist (comedy, 1930, pre-Code), 83,0 MB, URL:
https://archive.org/download/Golf_Specialist_1930/Golf_Specialist_512kb.mp4

Conspiracy (mystery melodrama, 1930, pre-Code), 642,2 MB, URL:
https://archive.org/download/Corspiracy/Corspiracy.mp4

Safe in Hell (thriller, 1931, pre-Code), 377,5 MB, URL:
https://archive.org/download/safe.in.hell.-1931.dsr.x-264-regret.mkv.mp-4-hd-wellman/safe.in.hell.1931.dsr.x264-regret.mkv.mp4%20%28%20HD%20%29%20Wellman.mp4
(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Night Nurse (crime drama mystery, 1931, pre-Code), 393,7 MB, URL:

https://archive.org/download/night-nurse-1931-wellman/Night%20Nurse%201931-%20Wellman.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

She Done Him Wrong (crime/comedy, 1933, pre-Code), 341,5 MB, URL:
https://archive.org/download/she-done-him-wrong-1933-restored-movie-720p-hd/she%20done%20him%20wrong-1933-restored%20movie-720p-hd.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Wild Boys of the Road (depression-era drama, 1933, pre-Code), 377,6 MB, URL:
https://archive.org/download/wild-boys-of-the-road-wellman/Wild%20Boys%20of%20the%20Road%20Wellman.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

The Man Who Knew Too Much (1934) (mystery and thriller, 1934), 271,1 MB, URL:
https://archive.org/download/the-man-who-knew-too-much-1934-hd-full-length-movie-directed-by-alfred-hitchcock/The%20Man%20Who%20Knew%20Too%20Much%20%281934%29%20HD%20Full%20Length%20Movie%20-%20Directed%20by%20Alfred%20Hitchcock.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Ninotchka (romantic comedy, 1939), 612 MB, URL:
https://archive.org/download/ninotchka-1939-restored-movie-720p-hd/ninotchka1939-restored%20movie-720p-hd-.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

The Maltese Falcon (film noir, 1941), 411,2 MB, URL:
https://archive.org/download/1941themaltesefalconelhalconmaltesjohnhuston/1941%20-%20The%20Maltese%20Falcon%20-%20El%20halc%C3%B3n%20malt%C3%A9s%20-%20John%20Huston%20-%20VOSE.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Sullivan's Travels (comedy, 1941), 498,5 MB, URL:
https://archive.org/download/sullivans-travels/Sullivan%27s%20Travels.mp4

A Close Call for Ellery Queen (mystery, 1942), 352,3 MB, URL:
https://archive.org/download/close-call-for-ellery-queen-1942/Close%20Call%20for%20Ellery%20Queen%20%281942%29%20.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Shadow of a Doubt (psychological thriller, 1943), 596,6 MB, URL:
https://archive.org/download/shadow-of-a-doubt-1943-restored-movie-720p-hd/shadow%20of%20a%20doubt-1943-restored%20movie-720p-hd.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Gaslight (psychological thriller, 1944), 275,2 MB, URL:
https://archive.org/download/Gaslight_628/Fanny_by_Gaslight_1945.mp4

Laura (film noir, 1944), mit portugiesischen Untertiteln, 522,9 MB, URL:
https://archive.org/download/laura-1944/Laura%20%281944%29.mp4

Spellbound (psychological thriller, 1945), 428,7 MB, URL:
https://archive.org/download/spellbound1945_202001/Spellbou
nd%20%281945%2C%20USA%29%20Director%20Alfred%20Hitchcock%20Star
ring%20Ingrid%20Bergman%2C%20Gregory%20Peck%20-%20Film%20Noir.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

The Strange Love of Martha Ivers (film noir, 1946), 481,4 MB, URL:
https://archive.org/download/Martha_Ivers_movie/Strange_Love_Mar
tha_Ivers_512kb.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Smart Girls Don't Talk (crime film, 1948), 448,2 MB, URL:
https://archive.org/download/smart-girls-dont-talk-1948-richard-l.-bare-virginia-mayo-bru
ce-bennett/Smart%20Girls%20Don%E2%80%99t%20Talk%20%281948%29%20Richa
rd%20L.%20Bare%2C%20Virginia%20Mayo%2C%20Bruce%20Bennett%2C.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Close-Up (film noir, 1948), 217,6 MB, URL:
https://archive.org/download/CloseUp_947/CloseUp1948.mp4

Kind Hearts and Coronets (crime black comedy, 1949), 561,3 MB, URL:
https://archive.org/download/kind-hearts-and-coronets/Kind%20Hea
rts%20and%20Coronets.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Quicksand (film noir, 1950), 437,2 MB, URL:
https://archive.org/download/quicksand-1950-remastered-movie-720p-hd/quick
sand-1950-remastered%20movie-720p-hd.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Cyrano de Bergerac (adventure comedy, 1950), 671,2 MB, URL:
https://archive.org/download/Cyrano_DeBergerac/Cyrano_De_Bergerac.mp4

The House on Telegraph Hill (film noir, 1951), 1,0 GB, URL:
https://archive.org/download/houseontelegraphhill1951/House%20on%20Tele
graph%20Hill%20%281951%2C%20USA%29%20Featuring%20Richard%20Base
hart%2C%20Valentina%20Cortese%20-%20Film%20Noir.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Cry Danger (film noir thriller, 1951), 351,8 MB, URL:
https://archive.org/download/cry-danger-1951/Cry%20Danger%201951%20-%20Di
ck%20Powell%2C%20Rhonda%20Fleming%2C%20Richard%20Erdman%2C%20Will
iam%20Conr_x264.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Man in the Attic (mystery, 1953), 454,6 MB, URL:
https://archive.org/download/man-in-the-attic-1953-restored-mo
vie-720p-hd/man%20in%20the%20attic-1953-restored%20movie-720p-hd.mp4

(Man beachte: Die URL wurde umgebrochen, damit sie auf die Druckbreite passt.)

Sabrina (romantic comedy-drama, 1954), mit portugies. Untertiteln, 675,1 MB, URL:
https://archive.org/download/sabrina-1954/Sabrina%20%281954%29.mp4

The Killing (film noir, 1956), 449,3 MB, URL:
https://archive.org/download/thekilling1956/The%20Killing%20%281956%29.mp4

Alfred Hitchcock Presents (season 2, episode 1, 1956), 139,3 MB, URL:
https://archive.org/download/alfred-hitchcock-presents-season-2/ahp-201.mp4

Alfred Hitchcock Presents (season 2, episode 2, 1956), 139,3 MB, URL:
https://archive.org/download/alfred-hitchcock-presents-season-2/ahp-201.mp4

„The Thin Man“ TV series: „Robot Client“ (season 1, episode 23, 1958), 195,5 MB, URL:
https://archive.org/download/TheThinManTvEpisode/TheThinManTvEpisode.mp4

„The Avengers“ TV series: „The Forget-Me-Knot“ (season 6, e. 1, 1968), 413,4 MB, URL:
https://archive.org/download/the-avengers-season-6-episodes-1-11/The%20Avengers%20-%20S06E01%20-%20The%20Forget-Me-Knot%20%2825%20September%201968%29.mp4

„The Avengers“ TV series: „Game“ (season 6, episode 2, 1968), 413,5 MB, URL:
https://archive.org/download/the-avengers-season-6-episodes-1-11/The%20Avengers%20-%20S06E02%20-%20Game%20%282%20October%201968%29.mp4

„The Avengers“ TV ser.: „Super Secret Cypher Snatch“ (s. 6, e. 3, 1968), 414,4 MB, URL:
https://archive.org/download/the-avengers-season-6-episodes-1-11/The%20Avengers%20-%20S06E03%20-%20Super%20Secret%20Cypher%20Snatch%20%289%20October%201968%29.mp4

Hörbücher in der Mediathek LibriVox

LibriVox bietet gemeinfreie Hörbücher im MP3-Format. Hier eine Auswahl in deutscher und englischer Sprache, mit verlustfreier Datenkompression (ZIP-Dateiformat). In der folgenden Liste wurden URL umgebrochen, damit sie auf die Druckbreite passen.

Tausendundeine Nacht, Band 3, 9. Jahrhundert (738 MB), URL:
https://www.archive.org/download//tausend_und_eine_nacht_band_3_1807_librivox/tausend_und_eine_nacht_band_3_1807_librivox_64kb_mp3.zip

Tausendundeine Nacht, Band 4, 9. Jahrhundert (557 MB), URL:
https://www.archive.org/download//tausend_und_eine_nacht_band_4_1903_librivox/tausend_und_eine_nacht_band_4_1903_librivox_64kb_mp3.zip

Charles Dickens: Dombey and Son, 1846 – 1848 (1,1 GB), URL:
https://www.archive.org/download/dombey_and_son_0901_librivox/dombey_and_son_0901_librivox_64kb_mp3.zip

Theodor Storm: Der Schimmelreiter, 1888 (123,6 MB), URL:
https://www.archive.org/download/schimmelreiter_storm_librivox/schimmelreiter_storm_librivox_64kb_mp3.zip

Jules Verne: Die Reise zum Mittelpunkt der Erde, 1864 (227 MB), URL:
https://www.archive.org/download//reise_nach_dem_mittelpunkt_der_erde_2012_librivox/reise_nach_dem_mittelpunkt_der_erde_2012_librivox_64kb_mp3.zip

Jules Verne: 20 000 Meilen unter dem Meer, 1869 – 1870 (410,1 MB), URL:
https://archive.org/download/20000_meilen_1001_li
brivox/20000_meilen_1001_librivox_64kb_mp3.zip

Karl May: Winnetou 1, 1893 (481,4 MB), siehe Seite 206, URL:
http://www.archive.org/download/winnetou1_librivox/winnetou1_librivox_64kb_mp3.zip

Karl May: Winnetou 2, 1893 (436 MB), URL:
https://www.archive.org/download/winne
tou_ii_2203_librivox/winnetou_ii_2203_librivox_64kb_mp3.zip

Kurd Laßwitz: Auf zwei Planeten, 1897 (695 MB), URL:
http://www.archive.org/download/auf_zwei_planeten_0907_librivox/auf_zwei_plane
ten_0907_librivox_64kb_mp3.zip

Ludwig Thoma: Tante Frieda (Neue Lausbubengeschichten), 1907 (78 MB), URL:
https://www.archive.org/download//tante_frieda_1606_libri
vox/tante_frieda_1606_librivox_64kb_mp3.zip

Fjodor Michailowitsch Dostojewski: Der Spieler, 1867 (168 MB), URL:
https://www.archive.org/download//spieler_1804_librivox/spieler_1804_li
brivox_64kb_mp3.zip

Theodor Mommsen: Römische Geschichte, erstes Buch, 1854 (439,3 MB), URL:
http://www.archive.org/download/roemische_geschichte_01_red_librivox/roemi
sche_geschichte_01_red_librivox_64kb_mp3.zip

Lewis Carroll: Alice's Adventures in Wonderland, 1865 (75,5 MB), URL:
http://www.archive.org/download/aliceinwonderland_1102_librivox/aliceinwonder
land_1102_librivox_64kb_mp3.zip

Sir Arthur Conan Doyle: The Adventures of Sherlock Holmes, 1892 (323,9 MB), URL:
http://www.archive.org/download/adventures_sherlockholmes_1007_librivox/adven
tures_sherlockholmes_1007_librivox_64kb_mp3.zip

Sir Arthur Ignatius Conan Doyle: Der Hund von Baskerville, 1901 – 1902 (197 MB), URL:
https://www.archive.org/download//hund_von_baskerville_1709_libri
vox/hund_von_baskerville_1709_librivox_64kb_mp3.zip

Matthias McDonnell Bodkin: Ein Münzverbrechen, 1897 (34,8 MiB), URL:
https://www.archive.org/download/sammlung_kurzer_deutscher_prosa_058_2210_li
brivox/sammlungprosa058_03_muenzverbrechen_sv_128kb.mp3

Montague Rhodes James: Ghost Stories of an Antiquary, 1904 (284,4 MB), URL:
https://www.archive.org/download/ghost_stories_antiquary_librivox/ghost_sto
ries_antiquary_librivox_64kb_mp3.zip

Fred M. White: The Yellow Face, 1907 (337 MB), URL:
https://www.archive.org/download/yellow_face_2112_librivox/yel
low_face_2112_librivox_64kb_mp3.zip

Arthur B. Reeve: The Dream Doctor, 1913 (293 MB), URL:

https://www.archive.org/download/dream_doctor_2201_librivox/dream_doctor_2201_librivox_64kb_mp3.zip

Burt L. Standish: Owen Clancy's Run of Luck, 1914 (61 MB), URL:
https://www.archive.org/download/owen_clancys_run_luck_hs_2206_librivox/owen_clancys_run_luck_hs_2206_librivox_64kb_mp3.zip

Earl Derr Biggers: The Agony Column, 1916 (70,6 MB), URL:
https://www.archive.org/download/agony_column_0902_librivox/agony_column_0902_librivox_64kb_mp3.zip

Dashiell Hammett: Zigzags of Treachery and other stories, 1924 (133 MB), URL:
https://www.archive.org/download/zigzags_treachery_wt_2202_librivox/zigzags_treachery_wt_2202_librivox_64kb_mp3.zip

G. K. Chesterton: The Incredulity of Father Brown, 1926 (199 MB), URL:
https://www.archive.org/download/incredulityfrbrown_2201_librivox/incredulityfrbrown_2201_librivox_64kb_mp3.zip

Clifford D. Simak: The World That Couldn't Be, 1958 (44 MB), URL:
https://www.archive.org/download/world_that-couldnt_be_1011_librivox/world_that-couldnt_be_1011_librivox_64kb_mp3.zip

Verschiedene Autoren: Short Science Fiction Collection 037, (207,5 MB), URL:
https://www.archive.org/download/short_scifi_037_1006_librivox/short_scifi_037_1006_librivox_64kb_mp3.zip

Verschiedene Autoren: Short Science Fiction Collection 086, (251 MB), URL:
https://www.archive.org/download/ssf_086_2206_librivox/ssf_086_2206_librivox_64kb_mp3.zip

H. Beam Piper: Five Sci-Fi Short Stories, 1948 – 1959 (144,5 MB), URL:
https://archive.org/download/five_sci-fi_piper_librivox/five_sci-fi_piper_librivox_64kb_mp3.zip

H. Beam Piper: The Cosmic Computer, 1963 (205 MB), URL:
http://www.archive.org/download/cosmic_computer_librivox/cosmic_computer_librivox_64kb_mp3.zip

Marion Zimmer Bradley: The Door Through Space, 1961 (108 MB), URL:
https://www.archive.org/download/door_through_space_0908/door_through_space_0908_64kb_mp3.zip

9.3 Sonstige Daten

Häufige Passwörter

Bei der Wahl von Passwörtern sollte man zumindest die am häufigsten verwendeten vermeiden. In Deutschland insbesondere: `123456`, `password`, `123456789`, `12345`, `hallo`, `passwort`, `ficken`, `12345678`, `master`, `1234`, `qwertz`, `hallo123`, `daniel`, `killer`, `123`, `111111`, `super123`, `guest`, `michael`, `matrix`.

Namen von Aktien für Python-Module yfinance und finanzen-fundamentals

Einige Symbole und WKN von Aktien für `yfinance` (siehe Seite 353) sind

```
"AFX.DE" : ["531370","Carl Zeiss Meditec   "],
"DTE.DE" : ["555750","Deutsche Telekom     "],
"HDD.DE" : ["731400","Heidelberger Druckm. "],
"JEN.DE" : ["A2NB60","Jenoptik             "],
"PFV.DE" : ["691660","Pfeiffer Vacuum Tec. "],
"QIA.DE" : ["A2DKCH","Qiagen               "],
"SNW.DE" : ["920657","Sanofi               "],
"SAP.DE" : ["716460","SAP SE               "],
"SRT3.DE": ["716563","Sartorius (Vorzüge)  "],
"SIE.DE" : ["723610","Siemens              "],
"SNG.DE" : ["A1681X","Singulus Technologies"],
```

Einige besondere Aktiennamen für `finanzen-fundamentals` (siehe Seite 354) sind

```
'adidas-aktie'                      : 'Adidas',      # WKN A1EWWW
'airbus-aktie'                      : 'Airbus',      # WKN 938914
'allianz-aktie'                     : 'Allianz',     # WKN 840400
'basf-aktie'                        : 'BASF',        # WKN BASF11
'bayer-aktie'                       : 'Bayer',       # WKN BAY001
'beiersdorf-aktie'                  : 'Beiersdorf',  # WKN 520000
'bmw-aktie'                         : 'BMW',         # WKN 519000
'brenntag-aktie'                    : 'Brenntag',    # WKN A1DAHH
'carl_zeiss_meditec-aktie'          : 'CZ Meditec',  # WKN 531370
'continental-aktie'                 : 'Continental', # WKN 543900
'covestro-aktie'                    : 'Covestro',    # WKN 606214
'daimler-aktie'                     : 'Daimler',     # WKN 710000
'delivery_hero-aktie'               : 'Deliv. Hero', # WKN A2E4K4
'deutsche_bank-aktie'               : 'Dt. Bank',    # WKN 514000
'deutsche_boerse-aktie'             : 'Dt. Börse',   # WKN 581005
'deutsche_post-aktie'               : 'Dt. Post',    # WKN 555200
'deutsche_telekom-aktie'            : 'Dt. Telekom', # WKN 555750
'eon-aktie'                         : 'E.ON',        # WKN ENAG99
'fresenius-aktie'                   : 'Fresenius  ', # WKN 578560
'fresenius_medical_care-aktie'      : 'Fres.M.Care', # WKN 578580
'heidelbergcement-aktie'            : 'Heid.Cement', # WKN 604700
'heidelberger_druckmaschinen-aktie' : 'Heid. Druck', # WKN 731400
'hellofresh-aktie'                  : 'HelloFresh',  # WKN A16140
'henkel-aktie'                      : 'Henkel',      # WKN 604840
'infineon-aktie'                    : 'Infineon',    # WKN 623100
'linde-aktie'                       : 'Linde',       # WKN A2DSYC
'merck-aktie'                       : 'Merck',       # WKN 659990
'mtu-aktie'                         : 'MTU Aero',    # WKN A0D9PT
'munich_re-aktie'                   : 'Münch. Rück', # WKN 843002
'porsche-aktie'                     : 'Porsche Vz.', # WKN PAH003
```

```
'puma-aktie'                    : 'Puma',        # WKN 696960
'qiagen-aktie'                  : 'Qiagen NV',   # WKN A2DKCH
'rwe-aktie'                     : 'RWE St.',     # WKN 703712
'sap-aktie'                     : 'SAP',         # WKN 716460
'sartorius_vz-aktie'            : 'Sartor. Vz.', # WKN 716563
'siemens-aktie'                 : 'Siemens',     # WKN 723610
'siemens_energy-aktie'          : 'S. Energy',   # WKN ENER6Y
'siemens_gamesa-aktie'          : 'S. Gamesa',   # WKN A0B5Z8
'siemens_healthineers-aktie'    : 'S. Health',   # WKN SHL100
'symrise-aktie'                 : 'Symrise',     # WKN SYM999
'volkswagen_vz-aktie'           : 'VW Vz.',      # WKN 766403
'vonovia-aktie'                 : 'Vonovia',     # WKN A1ML7J
'zalando-aktie'                 : 'Zalando',     # WKN ZAL111
```