

Linux – Gesamt-Lernheft

Grundlagen, Shell, Befehle, SSH, Rechte, Verzeichnisse, Pipe & Cron

🧭 So benutzt du das Lernheft

- **Links** ist das Inhaltsverzeichnis. Der Punkt davor zeigt die IHK-Relevanz.
- Jeder Abschnitt hat ein **IHK-Etikett** und einen farbigen Kasten mit der Begründung. Rot = sehr wichtig, Orange/Gelb = mittel, Grau = gering.
- **Gelb unterstrichen** = Prüfungs-Schlagwort. Das solltest du erklären können.
- Knopf **„🎯 Nur IHK-wichtig“** blendet alles mit „gering“ aus. Gut zum Wiederholen kurz vor der Prüfung.
- Bei jedem Abschnitt kannst du **„gelernt“** abhaken. Der Fortschritt wird im Browser gespeichert.
- Suche oben (Taste `/`), mit Enter springst du zum nächsten Treffer. Lösungen sind eingeklappt – erst selbst überlegen!

sehr hoch **hoch** **mittel bis hoch** **mittel** **gering bis mittel** **gering**

Die IHK-Einstufungen sind eigene Einschätzungen aus den Einzel-Zusammenfassungen, keine amtlichen Angaben der IHK.

🔥 Top-Prüfungsthemen

hoch 8 · **mittel bis hoch** 2 · **mittel** 28 · **gering bis mittel** 5 · **gering** 10 · **keine** 2

- **hoch** 2.2 📣 echo und Umleitung · 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug
- **hoch** 6.1 Was ist SSH? · 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug
- **hoch** 6.2 Voraussetzungen und Syntax · 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug
- **hoch** 2.1 📋 Grundlagen und ls -l lesen · 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug
- **hoch** 2.3 🧱 chmod – oktal und symbolisch · 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug
- **hoch** 2.5 🎭 umask – Standardrechte für neue Dateien · 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug
- **hoch** 2.7 🛡️ Principle of Least Privilege und Zero Trust · 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug
- **hoch** 5.1 🚰 Umleitung und Pipe · 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug

IHK	Abschnitt	Warum?
keine	1.1 📚 Die 9 Module des Kurses 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	nur Organisatorisches. Hilft dir beim Planen
keine	1.2 📖 Wo kann ich nachlesen und üben? 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	Hilfen zum Lernen, kein Prüfungsstoff
mittel	2.1 🖥️ Was ist ein Betriebssystem? 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	Betriebssysteme sind Thema in AP1 (“Arbeitsplatz konfigurieren”) und Grundlage für AP2 Konzeption/Administration (Einschätzung)
mittel	2.2 🧩 Merkmale von Linux 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	die Merkmale erklären, warum Linux auf Servern so verbreitet ist; gut für Kurzfragen in AP1 oder AP2 (Einschätzung)
gering	📜 Block 3: Geschichte – von Unix zu Linux 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	Jahreszahlen und Namen werden in der IHK-Prüfung kaum abgefragt. Wichtig ist nur der Kern: Linux ist ein Kernel und unixoid (Einschätzung)
gering bis mittel	4.1 🦬 Das GNU-Projekt 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	der Begriff GNU/Linux kann als Kurzfrage vorkommen; wichtiger ist der Unterschied Kernel / Werkzeuge (Einschätzung)
mittel	4.2 ⚖️ Open Source und freie Software 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	Lizenzen spielen bei der Auswahl von Software eine Rolle (AP1 “Software auswählen und Beschaffung einleiten”); ausdrückliche Lizenz-Aufgaben sind nicht belegt (Einschätzung)

IHK	Abschnitt	Warum?
mittel	5.1 📦 Was ist eine Distribution? 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	die passende Distribution für einen Einsatzzweck auswählen passt zu AP1 (“Software auswählen”) und AP2 Konzeption/Administration (Einschätzung)
mittel	5.2 🗂️ Wichtige Distributionen 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	Zuordnungen wie “Server mit langem Support → Ubuntu LTS” sind gut prüfbar (Einschätzung)
gering bis mittel	5.3 🌀 Debian 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	Debian ist das Kurssystem; die Eigenschaften “stabil, konservativ” sind wichtig, Codenamen kaum (Einschätzung)
mittel	6.1 📀 Installation in VirtualBox 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	ein System installieren und einrichten passt zu AP1 (“Arbeitsplatz konfigurieren und testen”); Virtualisierung ist in der Prüfungsübersicht nicht eigens genannt (Einschätzung)
mittel	6.2 🪟 Installation in Hyper-V 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	wie 6.1 (Einschätzung)
mittel bis hoch	6.3 ⌨️ Die ersten Befehle 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	Linux-Shell und Softwareinstallation gehören zum Linux-Stoff für AP2 Konzeption/Administration; Pakete aktualisieren ist Alltagsaufgabe (Einschätzung)
gering	6.4 🧭 Erkundungstour 🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug	Übungsaufgabe
mittel	1.1 Was ist eine Shell? 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	der Begriff Shell und ihre Rolle zwischen Benutzer und Kernel sind Grundwissen “Betriebssysteme” (AP1). Shell-Skripte sind Thema in AP2 Konzeption/Administration

IHK	Abschnitt	Warum?
gering	1.2 Verschiedene Shells 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Namen und Herkunft der Shells werden kaum abgefragt. Wichtig ist nur: Bash ist der Standard
mittel	1.3 Syntax-Schreibweise und Prompt 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	die Schreibweise <befehl> [optionen] [argumente] braucht man, um Aufgaben und Manpages richtig zu lesen
mittel	2.1 Übersicht 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Grundbefehle werden selten einzeln abgefragt. Sie sind aber Voraussetzung für jede Shell- und Skriptaufgabe in AP2 Konzeption/Administration
hoch	2.2 📣 echo und Umleitung 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Umleitung mit > und >> ist Grundlage für Skripte und Logdateien (AP2 Konzeption/Administration)
mittel	2.3 📂 ls – Verzeichnisinhalt auflisten 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	ls -l lesen ist Voraussetzung für Rechte-Aufgaben in AP2 Konzeption/Administration
mittel	2.4 📄 cat – Dateiinhalt anzeigen 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	häufig in Befehlsketten und beim Prüfen von Konfigurationsdateien
gering	2.5 👆 touch – Datei anlegen oder Zeitstempel setzen 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	nützliches Werkzeug, kaum eigene Prüfungsfrage

IHK	Abschnitt	Warum?
mittel	2.6 📋 cp, mv, rm – kopieren, verschieben, löschen 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Dateioperationen gehören zu jeder Administrationsaufgabe. rm ohne Papierkorb ist ein typischer Sicherheitshinweis
mittel	2.7 🗂️ mkdir, rmdir, cd – mit Verzeichnissen arbeiten 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Grundwissen für Pfadaufgaben und Skripte
mittel	3.1 🎛️ Optionen 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Optionen richtig lesen und schreiben ist Grundlage für Skriptaufgaben in AP2 Konzeption/Administration
mittel	3.2 🆘 Hilfe unter Linux 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	“Wie helfe ich mir selbst?” ist eine typische Praxisfrage. In AP1 passt sie zu Betriebssystemen und Kundeneinweisung
gering	4.1 ⏪ Historie, Suche, Vervollständigung 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Komfortfunktionen für die Praxis; selten Prüfungsstoff
gering	4.2 📜 less 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Werkzeug für die Praxis, z. B. beim Lesen von Logdateien
gering bis mittel	4.3 🏷️ alias 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Aliase zeigen den Umgang mit ~/.bashrc . Grundlage für die Anpassung der Arbeitsumgebung
gering	5.1 Überblick 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Editoren selbst werden kaum geprüft. Man braucht sie aber für jede Konfigurationsdatei und jedes Skript

IHK	Abschnitt	Warum?
gering	5.2 💾 Speichern und Beenden 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Praxiswissen. Ein Prüfer erwartet eher, dass man einen Editor sicher bedienen kann
hoch	6.1 Was ist SSH? 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	SSH ist das Standardprotokoll für sichere Fernwartung. Port 22 und der Telnet-Vergleich passen zu AP1 und AP2 Netzwerke
hoch	6.2 Voraussetzungen und Syntax 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Voraussetzungen einer Verbindung abprüfen ist eine typische Fehlersuche-Aufgabe
mittel	6.3 🔧 SSH-Server einrichten 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	Dienste installieren, starten und aktivieren ist Administrationspraxis (AP2 Konzeption/Administration)
mittel	6.4 🖧 Praxis: Von VM1 per SSH auf VM2 🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug	der Ablauf (IP ermitteln, Erreichbarkeit testen, verbinden, Fingerprint prüfen) zeigt systematische Fehlersuche
mittel	1.1 🔑 Root und sudo – die Idee 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	Rechtevergabe und Sicherheit sind Kern von AP2 Konzeption/Administration; in AP1 eher als allgemeines Sicherheitsprinzip
mittel	1.2 🛠️ sudo-Rechte vergeben 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	typisches Administrationsthema für AP2 Konzeption/Administration (“Benutzer- und Rechteverwaltung”)
gering	1.3 💲 Der Shell-Prompt 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	Grundwissen; hilft beim Lesen von Aufgaben, wird aber kaum einzeln abgefragt

IHK	Abschnitt	Warum?
hoch	2.1 📋 Grundlagen und ls -l lesen 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	Dateirechte sind Kernwissen für AP2 Konzeption/Administration; in AP1 möglich als Teil von “Betriebssysteme” und IT-Sicherheit
mittel	2.2 🔗 Hardlinks und symbolische Links 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	Unterschied Hardlink/Symlink ist ein klassisches Abgrenzungsthema für AP2 Konzeption/Administration
hoch	2.3 🧱 chmod – oktal und symbolisch 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	Oktalwerte umrechnen ist eine typische Kurzaufgabe mit Rechenweg; passt zu AP2 Konzeption/Administration
mittel	2.4 👤 chown und chgrp 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	gehört zur Benutzer- und Rechteverwaltung in AP2 Konzeption/Administration
hoch	2.5 🎭 umask – Standardrechte für neue Dateien 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	die umask-Rechnung ist eine gut prüfbare Rechenaufgabe für AP2 Konzeption/Administration
mittel bis hoch	2.6 ⭐ Spezialrechte: SetUID, SetGID, Sticky Bit 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	das Erkennen von s und t in ls -l ist eine typische Kurzfrage in AP2 Konzeption/Administration; Sicherheitsbezug
hoch	2.7 🛡️ Principle of Least Privilege und Zero Trust 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	IT-Sicherheitsprinzipien sind in AP1 (IT-Sicherheit am Arbeitsplatz) und in beiden AP2-Bereichen prüfbar

IHK	Abschnitt	Warum?
mittel	3.1 🌳 Verzeichnisbaum und FHS 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	Aufbau des Dateisystems ist Basiswissen “Betriebssysteme” (AP1) und Voraussetzung für AP2 Konzeption/Administration
mittel	3.2 🗄️ Standardverzeichnisse 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	Zuordnungsaufgaben (“Wo liegen Logdateien?”) sind gut prüfbar, v. a. AP2 Konzeption/Administration
gering bis mittel	3.3 👻 Virtuelle Verzeichnisse 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	eher Hintergrundwissen; /proc und IP-Forwarding können in AP2 Netzwerke oder Konzeption auftauchen
mittel	3.4 👉 Verzeichniszeiger und versteckte Dateien 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	und .. sind Voraussetzung für Pfadaufgaben; versteckte Dateien sind Basiswissen
mittel	4.1 📍 Absolute und relative Pfade 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	Pfadangaben tauchen in jeder Shell- oder Skriptaufgabe auf (AP2 Konzeption/Administration), selten als eigene Frage
gering	4.2 🏠 pwd und Tilde 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	Grundbefehle, die man können muss, die aber selten einzeln geprüft werden
hoch	5.1 🚰 Umleitung und Pipe 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	Befehlsketten sind Grundlage für Skripte und Automatisierung, die der Leitfaden für AP2 Konzeption/Administration ausdrücklich nennt
mittel	5.2 🔎 find 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	Dateisuche nach Kriterien ist typische Administrationspraxis in AP2 Konzeption/Administration

IHK	Abschnitt	Warum?
gering bis mittel	5.3 📇 locate 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	der Unterschied find/locate (live vs. Datenbank) ist eine gute Abgrenzungsfrage, sonst Randthema
mittel	5.4 ⏰ Cron 🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug	zeitgesteuerte Automatisierung passt zu “Programme zur automatisierten Systemverwaltung” in AP2 Konzeption/Administration

🐧 Linux Modul 1 – Zusammenfassung mit IHK-Bezug

📘 Grundlage: Folien “Linux, Modul 1: Grundlagen I” (GBS Fachschule, 22.10.2026). Die Zusammenfassung folgt den Themen der Folien.

- **Fett** gedruckte Begriffe sind Prüfungs-Schlagwörter. Diese Begriffe solltest du erklären können.
- 🎯 Jeder Abschnitt hat eine Zeile “IHK-Relevanz”. Sie ist eine eigene Einschätzung. Sie ist keine amtliche Angabe.
- Der IHK-Leitfaden nennt Linux nicht mit Namen. Linux passt aber zu diesen Prüfungsteilen:
  - **AP1** (Teil 1, “Einrichten eines IT-gestützten Arbeitsplatzes”): Betriebssysteme, Software auswählen und installieren.
  - **AP2 Konzeption und Administration**: Systeme planen, einrichten und verwalten.
- 💡 Modul 1 ist vor allem Grundwissen. Geschichte wird in der IHK-Prüfung kaum abgefragt. Begriffe wie Betriebssystem, Kernel, Distribution und Open Source sind dagegen wichtig.
- ✏️ Am Ende jedes Blocks stehen Übungsfragen. Die Lösungen stehen gesammelt am Ende.
- 🔎 Mit “Ergänzung” markierte Stellen stehen nicht auf den Folien. Sie sind aus Allgemeinwissen ergänzt und geprüft.

💡 Das Zeichen vorne in den Befehls-Beispielen heißt Prompt. `$` = normaler Benutzer. Den Prompt tippt man nicht mit.

Symbol	Bedeutung
🎯	IHK-Relevanz
💡	Merksatz
⚠️	Achtung, Falle
📜	Geschichte
⚖️	Lizenz und Rechte
📦	Pakete und Distributionen
💻	Praxis und Befehle
🔎	Ergänzung (nicht auf den Folien)
✏️	Übungsfragen
✅	Lösungen

🗺️ Block 1: Kursaufbau und Lernquellen

1.1 📚 Die 9 Module des Kurses IHK keine

> 🎯 **IHK: keine** nur Organisatorisches. Hilft dir beim Planen

Modul	Thema
1 bis 3	Linux-Grundlagen
4	Bootprozess
5	Festplatten und RAID
6	Paket-, Benutzer- und Prozessverwaltung
7	Shell Scripting
8	LAMP und Docker
9	Kali Linux und Passwortsicherheit

So läuft der Unterricht ab:

1. Erst kommt die Theorie.
2. Danach setzt man sie direkt praktisch um.
3. Übungsblätter vertiefen den Stoff.

📁 Folien und Material liegen online im Teams des Kurses.

1.2 📖 Wo kann ich nachlesen und üben? IHK keine

> 🎯 **IHK: keine** Hilfen zum Lernen, kein Prüfungsstoff

Bücher

Buch	Wofür gut
Linux kurz & gut	Nachschlagen der wichtigsten Befehle, für alle Distributionen
Linux – Der kompakte Grundkurs (Rheinwerk)	kompakter Einstieg
Linux – Das umfassende Handbuch (Rheinwerk)	das Standardwerk

Buch	Wofür gut
UNIX and Linux System Administration Handbook	Systemverwaltung (englisch)
The Linux Command Line	Einstieg Kommandozeile, Download unter linuxcommand.org

Zeitschriften: Linux-Magazin (für Admins und Entwickler), LinuxUser (für Alltagsnutzer), c’t (größte deutsche Computerzeitschrift), iX (professionelle IT).

Webseiten:

- Heise.de und Golem.de: IT-Nachrichten.
- LPI Learning (learning.lpi.org/de): Lernmaterial, auch für das Zertifikat **LPIC-1**.
- Debian Wiki (wiki.debian.org) und planet.debian.org: alles zu Debian.
- ubuntuusers.de: Hilfe zu Ubuntu und Linux.
- distrowatch.com: Infos und Ranglisten zu Linux- und BSD-Distributionen.
- wiki.archlinux.de: Anleitungen für Arch Linux.

💻 **Direkt üben:**

- UNIX Terminal Simulator (terminaltemple.com): Terminal im Browser.
- Mobile App “Linux Basics”.
- Linux Journey (labex.io/de/linuxjourney): interaktive Lernplattform.
- OverTheWire: Bandit: ein Spiel. Man löst Rätsel auf der Kommandozeile. Jedes Rätsel baut auf dem letzten auf.

✏️ Übungsfragen Block 1

Ü1.1 Nennen Sie zwei Quellen, mit denen Sie die Linux-Kommandozeile praktisch üben können.

Ü1.2 Für welches Linux-Zertifikat bietet LPI Learning Lernmaterial an?

🐧 Block 2: Was ist Linux?

2.1 🖥️ Was ist ein Betriebssystem? IHK mittel

> 🎯 **IHK: mittel** Betriebssysteme sind Thema in AP1 (“Arbeitsplatz konfigurieren”) und Grundlage für AP2 Konzeption/Administration (Einschätzung)

Stell dir ein Restaurant vor. Der Gast bestellt beim Kellner. Der Kellner bringt die Bestellung in die Küche. Der Gast geht nie selbst in die Küche. Beim Computer ist das **Betriebssystem** dieser

Kellner.

📖 Definition: Ein **Betriebssystem** ist eine Sammlung von Software. Es lässt Anwendungen laufen. Es verwaltet die **Systemressourcen**, also Rechenzeit, Speicher und Geräte.

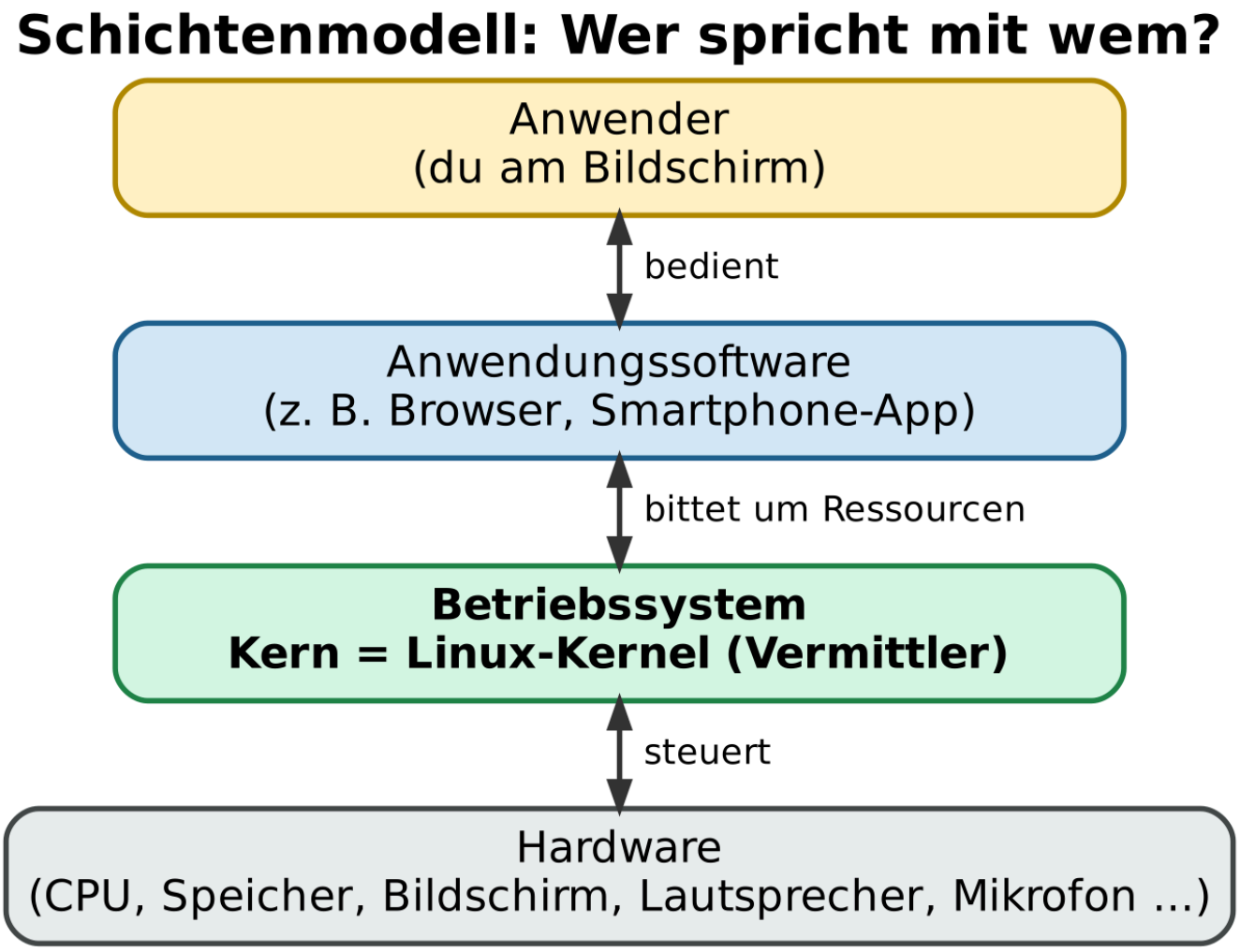
Beispiel Smartphone-App:

- Eine App läuft nicht allein. Sie braucht das Betriebssystem als Unterbau.
- Das Betriebssystem gibt der App Zugriff auf Internet, Bildschirm, Lautsprecher, Vibrationsmotor und Mikrofon.
- Erst so funktioniert die App, wie wir es kennen.

Ein Betriebssystem besteht aus vielen kleinen Teilen. Erst zusammen ergeben sie ein großes Ganzes.

💡 Merksatz: Der **Linux-Kernel** ist die zentrale Vermittlungsstelle zwischen Anwendungen und Hardware. **Kernel** heißt auf Deutsch “Kern”.

Schichtenmodell (von oben nach unten): Anwender → Anwendungssoftware → Betriebssystem → Hardware.



Schichtenmodell: Anwender bedient Anwendungssoftware, diese bittet das Betriebssystem um Ressourcen, der Linux-Kernel steuert die Hardware

Schichtenmodell: Anwender bedient Anwendungssoftware, diese bittet das Betriebssystem um Ressourcen, der Linux-Kernel steuert die Hardware

Bekannte Betriebssysteme (die Folie lässt die Liste leer; 🔎 Ergänzung): Windows, macOS, Linux, Android, iOS.

2.2 🧩 Merkmale von Linux IHK mittel

> 🎯 **IHK: mittel** die Merkmale erklären, warum Linux auf Servern so verbreitet ist; gut für Kurzfragen in AP1 oder AP2 (Einschätzung)

📖 Linux ist ein Betriebssystem. Genauer: Linux ist ein **unixoides System**. Das heißt: Es ist ähnlich wie Unix aufgebaut.

Merkmal	Einfach erklärt
Kernel + Userspace	Es gibt den Kern (Kernel). Daneben laufen die normalen Programme im Benutzerbereich (Userspace).
Starke Trennung	Kernel und Anwendungen sind klar getrennt.
Unix-Philosophie	Viele kleine Programme. Jedes tut eine Sache, aber die gut. Wie ein Werkzeugkasten statt eines Schweizer Messers.
Alles ist eine Datei	Auch Geräte, Prozesse und Netzwerke erscheinen als “Datei”.
Multiuser-System	Von Anfang an können mehrere Benutzer gleichzeitig arbeiten.
Klare Rechte	**root** ist der Administrator. Daneben gibt es normale Benutzer mit weniger Rechten.
Open Source	Der Quellcode ist frei verfügbar. Man kann das System anpassen und erweitern.

✏️ Übungsfragen Block 2

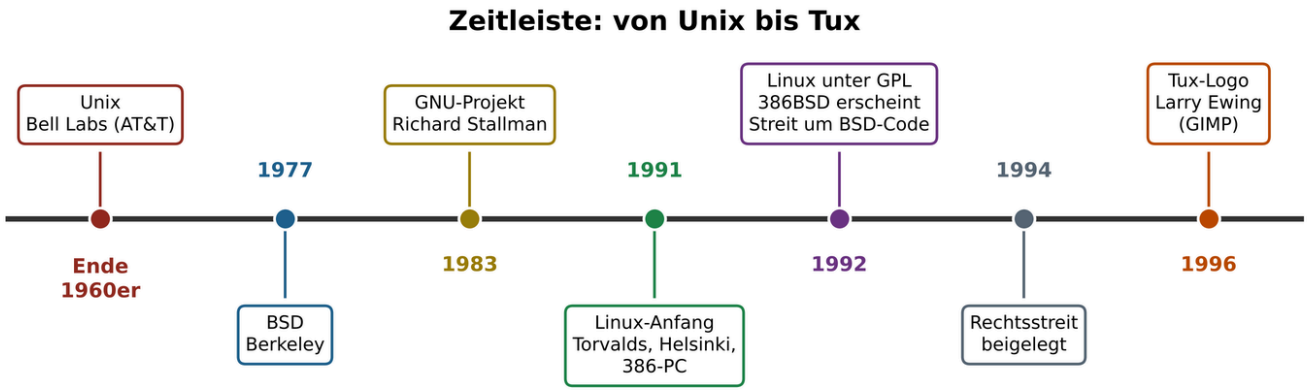
Ü2.1 Erklären Sie einem Kunden in zwei Sätzen, wozu ein Betriebssystem dient.

Ü2.2 Ordnen Sie die vier Schichten von oben nach unten: Hardware, Anwender, Betriebssystem, Anwendungssoftware.

Ü2.3 Was bedeutet “Unix-Philosophie”? Nennen Sie zwei weitere Merkmale eines Linux-Systems.

📜 Block 3: Geschichte – von Unix zu Linux IHK gering

> 🎯 **IHK: gering** Jahreszahlen und Namen werden in der IHK-Prüfung kaum abgefragt. Wichtig ist nur der Kern: Linux ist ein Kernel und unixoid (Einschätzung)



Zeitleiste von Unix (Ende 1960er) über BSD (1977), GNU (1983), Linux (1991), GPL und 386BSD (1992), Ende des Rechtsstreits (1994) bis zum Tux-Logo (1996)

Zeitleiste von Unix (Ende 1960er) über BSD (1977), GNU (1983), Linux (1991), GPL und 386BSD (1992), Ende des Rechtsstreits (1994) bis zum Tux-Logo (1996)

3.1 🏛️ Unix

- Unix ist eines der ältesten Betriebssysteme.
- Es entstand Ende der 1960er Jahre in den **Bell Laboratories** der Firma **AT&T**.
- Die Entwickler waren **Ken Thompson** und **Dennis Ritchie**.
- Ursprünglich lief Unix auf Großrechnern.
- Später entwickelten auch Universitäten Unix weiter.
- Bis Anfang der 1990er wurde Unix verkauft und in der Industrie breit genutzt.
- Viele Firmen hatten eine eigene Unix-Version, zum Beispiel Siemens, HP, Apple und IBM.

3.2 🌆 BSD

- **1977** entstand an der Universität von Kalifornien in **Berkeley** ein eigenes Unix: **BSD**.
- BSD steht für Berkeley Software Distribution.
- BSD ist eine Abspaltung von Unix. Es ist sehr ähnlich, ging aber eigene Wege.

Was war bei BSD neu?

- ⚖️ BSD war von Anfang an frei verfügbar. Es steht unter der **BSD-Lizenz**. Das kommerzielle Unix war dagegen kostenpflichtig.
- Aus BSD stammen freie Systeme: **FreeBSD**, **OpenBSD** und **NetBSD**.
- BSD brachte neue Funktionen:
  - **TCP/IP**-Netzwerk (die Grundlage des Internets),
  - das verbesserte Dateisystem **FFS**,

    - viele Werkzeuge, die heute Standard sind, z. B. **vi** (Texteditor) und **grep** (sucht Textmuster in Dateien).

🔎 Ergänzung: grep wurde ursprünglich im Unix der Bell Labs entwickelt. BSD hat es übernommen und weit verbreitet. vi dagegen entstand in Berkeley.

Quizfragen der Folien (die Folien nennen keine Antworten; 🔎 Ergänzung, geprüft):

- “Was ist Orbis OS?” Das Betriebssystem der Spielkonsole PlayStation 4 von Sony. Es basiert auf FreeBSD.
- “Was ist Darwin?” Der Unix-artige Kern der Apple-Systeme (macOS, iOS). Darwin enthält Teile von BSD.

💡 Merksatz: BSD-Code steckt in vielen Geräten, die man nicht als “Unix” erkennt.

3.3 🐣 Wie Linux entstand

1991: **Linus Torvalds** studierte an der Universität **Helsinki**. Er begann Linux als Hobbyprojekt.

Der Anlass: Er hatte einen neuen **386-PC**. Er wollte dessen Möglichkeiten besser nutzen. Dafür wollte er ein Unix-ähnliches System.

Das Problem damals:

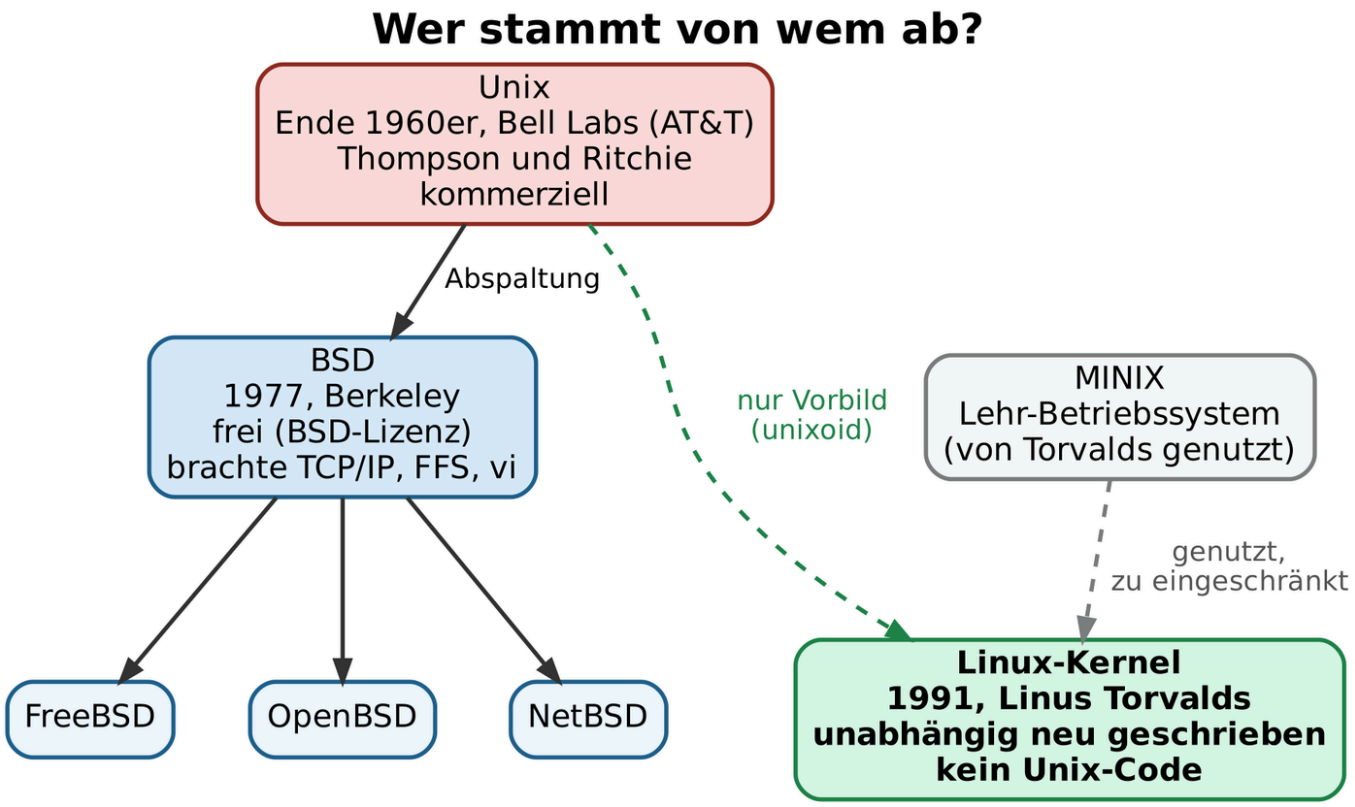
- Kommerzielles Unix war teuer.
- **MINIX** war ein einfaches Lehr-Betriebssystem. Torvalds nutzte es. Es hatte aber nur wenige Funktionen.
- Eine gute, freie Alternative für den PC fehlte.

Die Lösung: Torvalds experimentierte mit der Hardware. Daraus entwickelte er Schritt für Schritt einen eigenen Kernel.

Das Ergebnis:

- offen, kostenlos und frei verfügbar,
- von einer Gemeinschaft (Community) weiterentwickelt,
- läuft auf normaler PC-Hardware.

⚠️ Wichtig für das Verständnis: **Linux war zuerst nur ein Kernel.** Es war kein vollständiges Betriebssystem. Viele andere Programme gab es schon, z. B. vom GNU-Projekt. Es fehlte aber ein freier Kernel. Torvalds hatte anfangs keinen großen Plan dafür.



Stammbaum: Aus Unix spaltet sich BSD ab, daraus FreeBSD, OpenBSD und NetBSD. Linux ist unabhängig neu geschrieben, Unix war nur Vorbild, MINIX wurde genutzt.

Stammbaum: Aus Unix spaltet sich BSD ab, daraus FreeBSD, OpenBSD und NetBSD. Linux ist unabhängig neu geschrieben, Unix war nur Vorbild, MINIX wurde genutzt.

💡 Merksatz: Linux ist unixoid, enthält aber keinen Unix-Code. Torvalds schrieb den Kernel unabhängig.

3.4 🤔 Warum nicht einfach BSD?

Die Lage 1991:

- Vollständige BSD-Versionen enthielten noch Unix-Code. Für diesen Code brauchte man eine Lizenz.
- Freie BSD-Teile gab es. Ein vollständiges freies BSD für den 386-PC gab es aber nicht.
- **386BSD** (freies BSD für 386-PCs) erschien erst **1992**. Da lief Linux schon.

Dazu kam ab 1992 ein **Rechtsstreit**:

- AT&T behauptete: BSD enthält geschützten Unix-Code. Dieser darf nicht ohne Erlaubnis verbreitet werden.
- Niemand wusste sicher, ob man BSD bedenkenlos nutzen durfte.
- Der Streit wurde erst **1994** beigelegt.

Linux war unabhängig entwickelt. Der Streit betraf Linux nicht. Darum verbreitete sich Linux schnell.

3.5 🐧 Tux, das Maskottchen

- Das Linux-Logo ist ein glücklicher Pinguin namens **Tux**.
- Die Idee hatte Torvalds nach einem Zoobesuch.
- Tux soll aussehen wie ein zufriedener Pinguin nach einem Bier.
- **Larry Ewing** zeichnete ihn **1996** mit dem Grafikprogramm **GIMP**.

3.6 ⚖️ Linux wird Open Source

- **1992** erschien Linux unter der **GNU General Public License (GPL)**.
- Seitdem darf jeder Linux frei nutzen, verändern und weitergeben. Dabei gelten die Bedingungen der GPL.
- Folge: Die Community entwickelte Linux schnell weiter.

✏️ Übungsfragen Block 3

Ü3.1 Warum sagt man: “Linux war am Anfang kein vollständiges Betriebssystem”?

Ü3.2 Nennen Sie zwei Gründe, warum sich 1991/1992 Linux und nicht BSD auf dem PC durchsetzte.

Ü3.3 Ein Kollege sagt: “Linux ist eine Kopie von Unix-Code.” Nehmen Sie Stellung.

🆓 Block 4: GNU und Open Source

4.1 🦬 Das GNU-Projekt IHK gering bis mittel

> 🎯 **IHK: gering bis mittel** der Begriff GNU/Linux kann als Kurzfrage vorkommen; wichtiger ist der Unterschied Kernel / Werkzeuge (Einschätzung)

- **GNU** steht für **“GNU’s Not Unix”**.
- Das ist ein **rekursives Akronym**: Die Abkürzung steckt in ihrer eigenen Erklärung.
- **Richard Stallman** gründete GNU **1983**.
- Ziel: ein vollständig freies, Unix-ähnliches Betriebssystem.

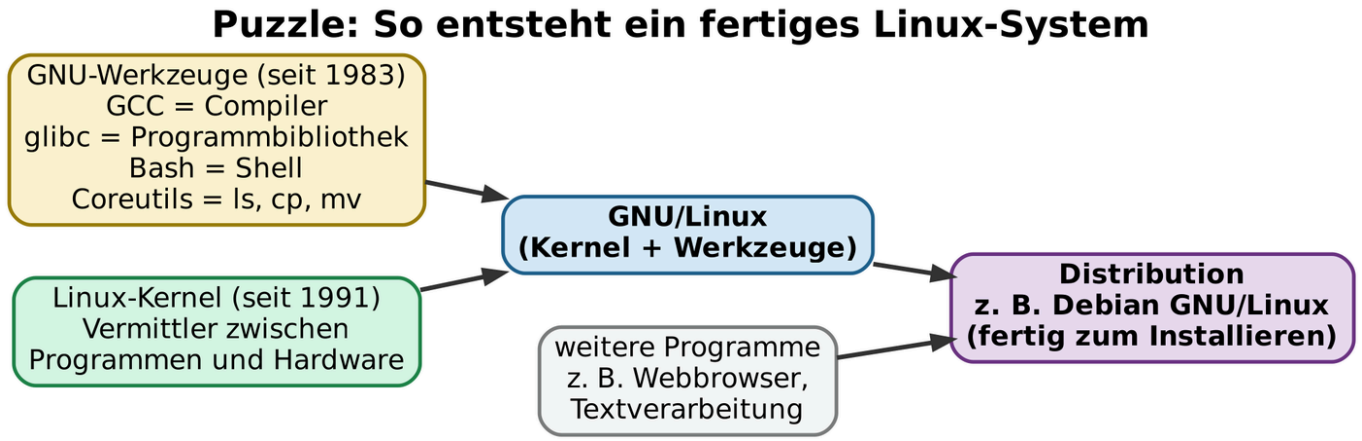
⚖️ Philosophie **freier Software**: Nutzer dürfen Software frei **verwenden, untersuchen, verändern und weitergeben**.

Wichtige GNU-Bausteine:

Baustein	Was ist das?
GCC	GNU Compiler Collection. Übersetzt Programmcode verschiedener Sprachen in ausführbare Programme.
glibc	GNU C Library. Grundlegende Programmbibliothek, die viele Programme nutzen.
Bash	Bourne Again Shell. Die Kommandozeile, in die man Befehle tippt.
GNU Coreutils	Grundbefehle wie `ls`, `cp` und `mv`.

Wie ein Puzzle: GNU lieferte fast alle Teile. Es fehlte der Kernel. Linux war das fehlende Teil.

💡 Merksatz: **GNU-Werkzeuge + Linux-Kernel = GNU/Linux.** Darauf bauen viele Distributionen auf, z. B. Debian GNU/Linux.



Puzzle: GNU-Werkzeuge und Linux-Kernel ergeben GNU/Linux; zusammen mit weiteren Programmen entsteht eine Distribution

Puzzle: GNU-Werkzeuge und Linux-Kernel ergeben GNU/Linux; zusammen mit weiteren Programmen entsteht eine Distribution

4.2 ⚖️ Open Source und freie Software IHK mittel

> 🎯 **IHK: mittel** Lizenzen spielen bei der Auswahl von Software eine Rolle (AP1 “Software auswählen und Beschaffung einleiten”); ausdrückliche Lizenz-Aufgaben sind nicht belegt (Einschätzung)

Linux ist **Open-Source-Software**. Das heißt:

- Jeder darf den **Quellcode** einsehen.
- Jeder darf ihn verändern.
- Jeder darf zur Weiterentwicklung beitragen.
- Private **und** kommerzielle Nutzung sind erlaubt.
- Weitergeben ist erlaubt. Verteilen ist sogar ausdrücklich erwünscht.

⚠️ Falle: “Frei” heißt nicht “kostenlos”. Stallman sagt dazu:

> **“Free Software – free as in freedom, not as free beer.”**

Gemeint ist die Freiheit, Software zu nutzen, zu verändern und weiterzugeben. Es geht nicht um “Freibier”.



Die vier Freiheiten freier Software: verwenden, untersuchen, verändern, weitergeben. Free bedeutet Freiheit, nicht kostenlos.

Die vier Freiheiten freier Software: verwenden, untersuchen, verändern, weitergeben. Free bedeutet Freiheit, nicht kostenlos.

Warum ist Open Source wichtig? (laut Folien)

- **Offenheit und Zusammenarbeit**: Wissen wird geteilt. Jeder kann lernen, verstehen und verbessern.
- **Gemeinschaft statt Konkurrenz**: Viele helfen freiwillig. Fehler werden gemeinsam gefunden.
- **Wertschätzung und Verantwortung**: Freiwillige Arbeit verdient Anerkennung. Lizenzbedingungen muss man einhalten.
- **Mitmachen statt nur nutzen**: mit Code, Dokumentation, Tests oder Spenden.

Bekannte Open-Source-Software (die Folie lässt die Liste leer; 🔎 Ergänzung): Linux-Kernel, Firefox, LibreOffice, VLC, GIMP, Apache, MySQL/MariaDB.

Wo ist Linux im Einsatz?

- Aus den Folien: Server und Unternehmenssysteme (Ubuntu LTS), Einplatinencomputer (Raspberry Pi OS), Penetrationstests (Kali Linux).
- 🔎 Ergänzung: Smartphones (Android nutzt den Linux-Kernel), Router, Embedded-Geräte, Supercomputer.

✏️ Übungsfragen Block 4

Ü4.1 Wofür steht GNU, und was ist an dieser Abkürzung besonders?

Ü4.2 Ein Kunde sagt: “Open Source heißt doch nur, dass es nichts kostet.” Korrigieren Sie ihn.

Ü4.3 Was ist der Unterschied zwischen “Linux” und “GNU/Linux”?

📦 Block 5: Distributionen und Debian

5.1 📦 Was ist eine Distribution? IHK mittel

> 🎯 **IHK: mittel** die passende Distribution für einen Einsatzzweck auswählen passt zu AP1 (“Software auswählen”) und AP2 Konzeption/Administration (Einschätzung)

Den Linux-Kernel allein kann man nicht sinnvoll nutzen. Man installiert Linux deshalb über eine **Distribution**.

Vergleich: Der Kernel ist der Motor. Eine Distribution ist das fertige Auto. Es gibt Sportwagen, Lieferwagen und Familienautos. Alle haben einen Motor, aber jedes ist für etwas anderes gebaut.

📖 Definition: Eine **Distribution** ist eine **Sammlung von Softwarepaketen**. Sie enthält den Linux-Kernel und weitere Programme, z. B. Webbrowser und Textverarbeitung.

- Jede Distribution ist für bestimmte Einsatzgebiete optimiert. Beispiele: besonders stabile Software oder lange Unterstützung.
- Jede folgt eigenen Ideen und Entscheidungen. Darum gibt es so viele verschiedene.

5.2 🗂️ Wichtige Distributionen IHK mittel

> 🎯 **IHK: mittel** Zuordnungen wie “Server mit langem Support → Ubuntu LTS” sind gut prüfbar (Einschätzung)

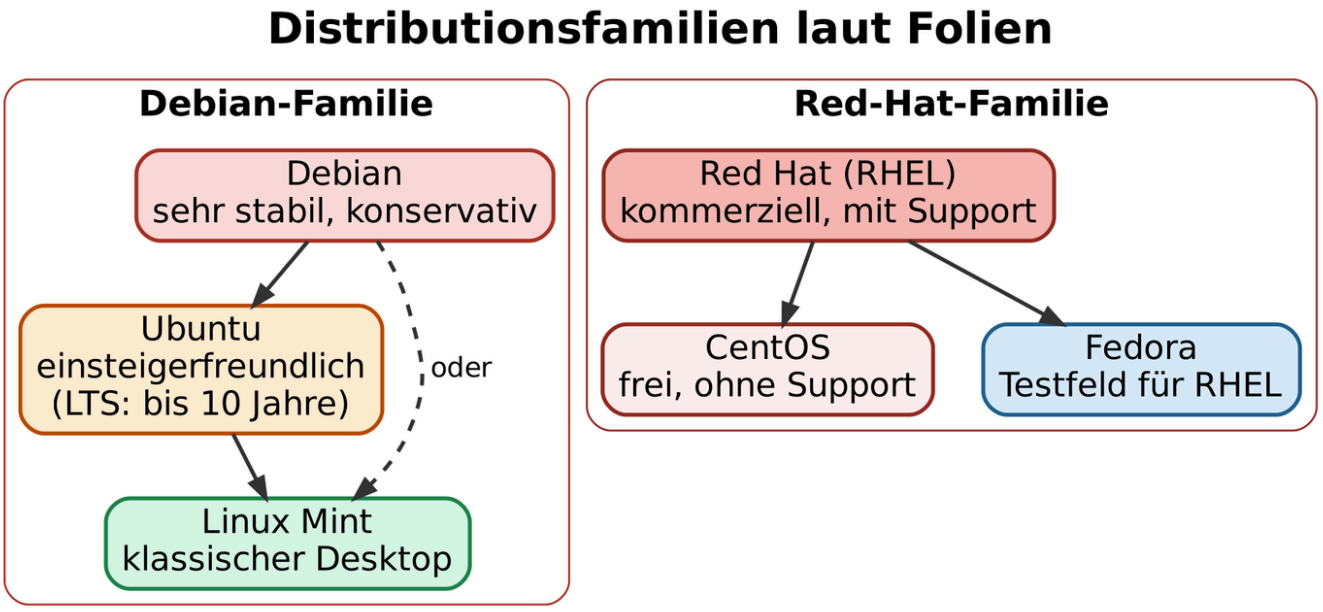
Distribution	Kennzeichen
Debian	sehr alt und sehr stabil, strenge Paketpolitik, Basis vieler anderer Systeme
Ubuntu	benutzerfreundlich, gut für Einsteiger, Debian-Basis, große Community

Distribution	Kennzeichen
Ubuntu LTS	**LTS = Long Term Support**: bis zu 10 Jahre Unterstützung, beliebt für Server und Firmen
Linux Mint	benutzerfreundlich, Basis Ubuntu oder Debian, klassischer Desktop
Red Hat (RHEL)	kommerziell für Unternehmen: Stabilität, Support, zertifizierte Lösungen
CentOS	freie Community-Variante von Red Hat Enterprise Linux, stabil, ohne kommerziellen Support
Fedora	modern, von Red Hat und Community, bringt neue Technik früh, Testfeld für RHEL
SUSE Linux	deutsche Unternehmens-Distribution, Werkzeug **YaST** (Yet another Setup Tool) zum Einrichten
Arch Linux	immer die neuesten Pakete, minimalistisch, **Rolling Release**, volle Kontrolle
Kali Linux	spezialisiert auf Penetrationstests und IT-Sicherheit
Raspberry Pi OS	für Einplatinencomputer und Bastelprojekte
Gentoo	quellbasiert: Programme werden aus dem Quellcode selbst übersetzt (kompiliert)
Slackware	eine der ältesten aktiven Distributionen, einfach, stabil, nah an der Unix-Philosophie
Linux from Scratch	keine fertige Distribution, sondern eine Anleitung: Linux selbst aus dem Quellcode bauen

📖 Zwei Begriffe einfach erklärt:

- **LTS (Long Term Support)**: Eine Version bekommt sehr lange Sicherheits-Updates. Gut für Server, die jahrelang laufen.
- **Rolling Release**: Es gibt keine großen Versionssprünge. Das System wird laufend aktualisiert. Wie eine Zeitung, die ständig neu gedruckt wird.

🔎 Ergänzung zu CentOS: Das klassische CentOS Linux wurde inzwischen eingestellt. Heute gibt es CentOS Stream.



Distributionsfamilien: Debian ist Basis für Ubuntu, Ubuntu oder Debian für Linux Mint; Red Hat ist Basis für CentOS und Fedora

Distributionsfamilien: Debian ist Basis für Ubuntu, Ubuntu oder Debian für Linux Mint; Red Hat ist Basis für CentOS und Fedora

💡 Merksatz: Debian → Ubuntu → Mint. Red Hat → CentOS und Fedora.

ℹ️ Die Grafik zur Entwicklung der Distributionen auf den Folien reicht nur bis 2006. Neuere Übersichten findet man online unter “Linux Distribution Timeline”.

5.3 🌀 Debian IHK gering bis mittel

> 🎯 **IHK: gering bis mittel** Debian ist das Kurssystem; die Eigenschaften “stabil, konservativ” sind wichtig, Codenamen kaum (Einschätzung)

Eigenschaften von Debian:

- weit verbreitet, in vielen Bereichen im Einsatz,
- riesige Auswahl an Software,
- viel Dokumentation und eine große Online-Community,
- **konservative Entwicklung**: Software wird gründlich getestet, bevor sie erscheint. Lieber etwas älter, dafür zuverlässig.

Codenamen: Jede Debian-Version heißt nach einer Figur aus den Toy-Story-Filmen.

Version	Codename	Figur
1.1	Buzz	Raumfahrer Buzz Lightyear
1.2	Rex	grüner Dinosaurier
1.3	Bo	Schäferin Bo Peep

Version	Codename	Figur
2.0	Hamm	Sparschwein
4.0	Etch	Zeichentafel “Etch A Sketch”
6.0	Squeeze	kleine grüne dreiäugige Aliens
7	Wheezy	kleiner Pinguin mit roter Fliege
8	Jessie	Cowgirl
9	Stretch	lila Oktopus
10	Buster	Andys Hund
11	Bullseye	Woodys Pferd
12	Bookworm	kleiner grüner Bücherwurm
13	Trixie	blauer Triceratops

ℹ️ Die Liste der Folien ist nicht vollständig. Es fehlen z. B. die Versionen 3.x und 5.0.

✏️ Übungsfragen Block 5

Ü5.1 Erklären Sie den Begriff “Distribution” in einem Satz.

Ü5.2 Ein Kunde braucht einen Server, der viele Jahre ohne großen Umbau laufen soll. Welche Distribution aus den Folien empfehlen Sie, und warum?

Ü5.3 Was ist der Unterschied zwischen einem LTS-Modell und einem Rolling Release?

💻 Block 6: Praxis – Debian installieren und aktualisieren

6.1 📀 Installation in VirtualBox IHK mittel

> 🎯 **IHK: mittel** ein System installieren und einrichten passt zu AP1 (“Arbeitsplatz konfigurieren und testen”); Virtualisierung ist in der Prüfungsübersicht nicht eigens genannt (Einschätzung)

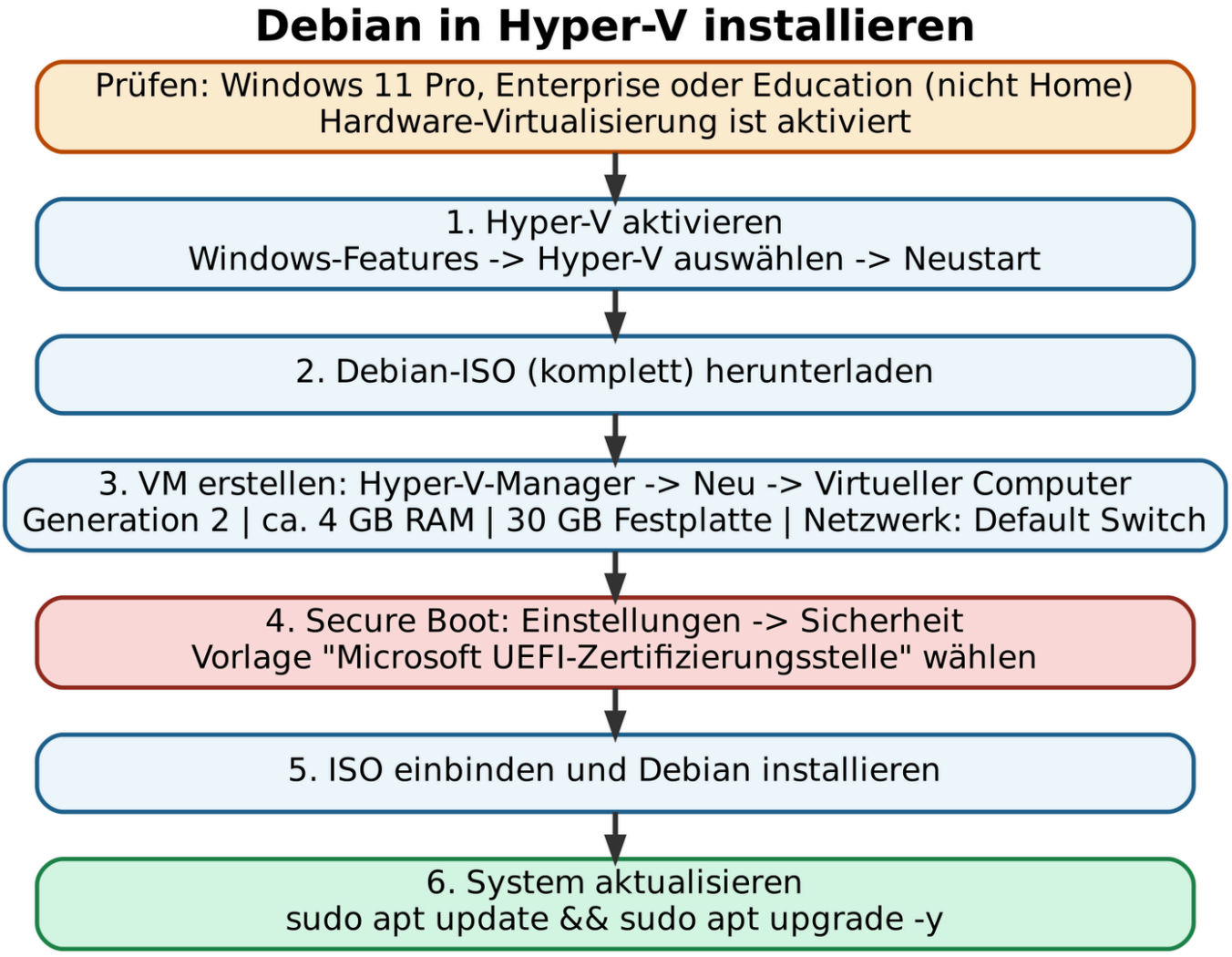
Eine **virtuelle Maschine (VM)** ist ein Computer im Computer. Sie läuft als Programm auf deinem PC. So kannst du Linux gefahrlos ausprobieren.

1. VirtualBox installieren (virtualbox.org/wiki/Downloads).
2. Eventuell ist “Microsoft Visual C++ 2019 Redistributable” nötig.
3. Debian-ISO herunterladen (debian.org/distrib/). Wichtig: aktuelle stabile Version, Architektur **amd64**.
4. Neue virtuelle Maschine anlegen.
5. Debian installieren. Anleitung: Debian Handbook (debian-handbook.info), Kapitel Installationsschritte.
6. System aktualisieren (siehe 6.3).
7. Optional: grafische Oberfläche installieren.

6.2 🪟 Installation in Hyper-V IHK mittel

> 🎯 **IHK: mittel** wie 6.1 (Einschätzung)

⚠️ Voraussetzung: **Windows 11 Pro, Enterprise oder Education**. Mit Windows 11 **Home** geht es nicht. Außerdem muss die Hardware-Virtualisierung aktiviert sein.



Ablauf der Installation in Hyper-V: Voraussetzungen prüfen, Hyper-V aktivieren, ISO laden, VM erstellen, Secure Boot einstellen, installieren, aktualisieren

Ablauf der Installation in Hyper-V: Voraussetzungen prüfen, Hyper-V aktivieren, ISO laden, VM erstellen, Secure Boot einstellen, installieren, aktualisieren

Die Einstellungen der VM im Überblick:

Einstellung	Wert
Generation	**Generation 2**
Arbeitsspeicher	ca. **4 GB RAM**
Festplatte	**30 GB**
Netzwerk	**Default Switch**
Secure Boot	Vorlage **“Microsoft UEFI-Zertifizierungsstelle”**

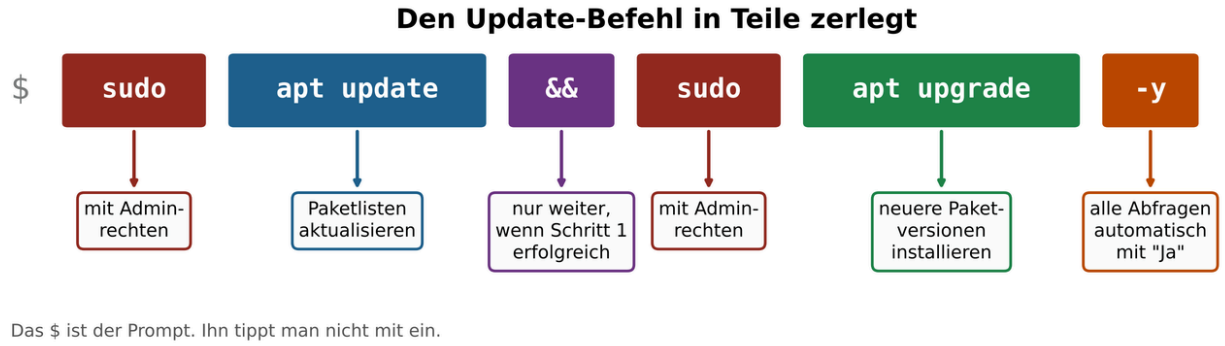
⚠️ Falle: Ohne die richtige Secure-Boot-Vorlage startet das Debian-ISO in der VM meist nicht. Man stellt sie unter Einstellungen → Sicherheit ein.

6.3 ⌨️ Die ersten Befehle IHK mittel bis hoch

> 🎯 **IHK: mittel bis hoch** Linux-Shell und Softwareinstallation gehören zum Linux-Stoff für AP2 Konzeption/Administration; Pakete aktualisieren ist Alltagsaufgabe (Einschätzung)

System aktualisieren:

```
$ sudo apt update && sudo apt upgrade -y
```



Der Update-Befehl in Teile zerlegt: sudo, apt update, &&, sudo, apt upgrade, -y

Der Update-Befehl in Teile zerlegt: sudo, apt update, &&, sudo, apt upgrade, -y

Teil	Bedeutung
`sudo`	führt den Befehl mit Administratorrechten aus (🔎 Ergänzung)

Teil	Bedeutung
`apt`	die **Paketverwaltung** von Debian: Sie installiert und aktualisiert Software
`apt update`	lädt die aktuellen **Paketlisten**. Es installiert noch nichts. (🔎 Ergänzung)
`apt upgrade`	installiert neuere Versionen der schon installierten Pakete (🔎 Ergänzung)
`&&`	verbindet zwei Befehle. Der zweite läuft nur, wenn der erste erfolgreich war. (🔎 Ergänzung)
`-y`	steht für “yes”. Beantwortet die Bestätigungsfragen automatisch mit “Ja”.

Vergleich zu `update` und `upgrade`: `update` ist wie der neue Katalog im Briefkasten. `upgrade` ist die eigentliche Bestellung der neuen Sachen.

⚠️ Falle: Nur `apt update` reicht nicht. Dann kennt das System die neuen Versionen, hat sie aber noch nicht installiert.

Grafische Oberfläche installieren (optional):

```
$ sudo apt install task-gnome-desktop -y
```

- `apt install <Paket>` installiert ein Paket (🔎 Ergänzung).
- Hier ist es das Paket `task-gnome-desktop`. Das ist die grafische Oberfläche GNOME.

Befehle, die in den Folien nur mit Namen vorkommen:

Befehl	Herkunft	Was er tut
`ls`	GNU Coreutils	Inhalt eines Verzeichnisses auflisten (🔎 Ergänzung)
`cp`	GNU Coreutils	Dateien kopieren (🔎 Ergänzung)
`mv`	GNU Coreutils	Dateien verschieben oder umbenennen (🔎 Ergänzung)
`vi`	BSD	Texteditor
`grep`	laut Folien BSD	sucht Textmuster in Dateien

6.4 🧭 Erkundungstour IHK gering

> 🎯 **IHK: gering** Übungsaufgabe

Probiere einen dieser beiden Links aus:

- LPIC-1 Exam 101 Lernmaterial (learning.lpi.org/de/learning-materials/101-500/)

- Debian Reference, Kapitel 1 “GNU/Linux-Lehrstunde” (debian.org/doc/manuals/debian-reference/ch01.de.html)

💡 Merksatz der Folien: “Keine Angst vor der Kommandozeile – sie ist Ihr bester Freund im Linux-Universum!”

✏️ Übungsfragen Block 6

Ü6.1 Ein Kollege will Debian in Hyper-V auf einem Laptop mit Windows 11 Home installieren. Was sagen Sie ihm?

Ü6.2 Erklären Sie jeden Teil des Befehls `sudo apt update && sudo apt upgrade -y`.

Ü6.3 Was passiert, wenn `apt update` fehlschlägt? Läuft `apt upgrade` trotzdem?

✅ Lösungen

Block 1

Ü1.1 Zwei von diesen: UNIX Terminal Simulator (terminaltemple.com), App “Linux Basics”, Linux Journey, OverTheWire: Bandit.

Ü1.2 Für **LPIC-1**.

Block 2

Ü2.1 Ein Betriebssystem lässt Anwendungen laufen. Es verwaltet die Systemressourcen und vermittelt zwischen Programmen und Hardware.

Ü2.2 Anwender → Anwendungssoftware → Betriebssystem → Hardware.

Ü2.3 Unix-Philosophie: Viele kleine Programme tun jeweils eine Sache gut. Weitere Merkmale (zwei davon): Alles ist eine Datei; Multiuser-System; klare Rechte (root und normale Nutzer); Open Source; starke Trennung von Kernel und Anwendungen.

Block 3

Ü3.1 Torvalds schrieb zuerst nur den Kernel. Die übrigen Programme, z. B. Shell und Grundbefehle, kamen von anderen Projekten wie GNU. Erst zusammen entsteht ein vollständiges System.

Ü3.2 Zwei von diesen:

- 1991 gab es kein vollständiges freies BSD für den 386-PC. 386BSD kam erst 1992.
- Vollständige BSD-Versionen enthielten noch lizenzpflichtigen Unix-Code.

- Ab 1992 gab es einen Rechtsstreit um BSD-Code (beigelegt 1994). Linux war davon nicht betroffen.

Ü3.3 Die Aussage ist falsch. Linux ist **unixoid**, also nur ähnlich aufgebaut wie Unix. Torvalds schrieb den Kernel unabhängig. Linux enthält keinen Unix-Code. Darum war es vom Rechtsstreit um BSD nicht betroffen.

Block 4

Ü4.1 GNU = “GNU’s Not Unix”. Es ist ein rekursives Akronym: Die Abkürzung GNU kommt in ihrer eigenen Auflösung wieder vor.

Ü4.2 “Free” bedeutet Freiheit, nicht kostenlos (“free as in freedom, not as free beer”). Man darf die Software verwenden, untersuchen, verändern und weitergeben. Dabei gelten Lizenzbedingungen, z. B. die GPL.

Ü4.3 “Linux” ist genau genommen nur der Kernel. “GNU/Linux” ist der Linux-Kernel zusammen mit den GNU-Werkzeugen, z. B. GCC, glibc, Bash und Coreutils. Das ergibt ein nutzbares Betriebssystem.

Block 5

Ü5.1 Eine Distribution ist eine Sammlung von Softwarepaketen aus Linux-Kernel und weiteren Programmen, optimiert für einen bestimmten Einsatzzweck.

Ü5.2 Zum Beispiel **Ubuntu LTS**: Es bietet bis zu 10 Jahre Support und ist laut Folien beliebt für Server und Unternehmen. Auch Debian (sehr stabil, gründlich getestet) oder Red Hat Enterprise Linux (mit kommerziellem Support) sind gut begründbar.

Ü5.3 LTS: Eine Version bekommt viele Jahre lang Updates. Große neue Versionen kommen selten. Rolling Release (z. B. Arch Linux): Es gibt keine festen Versionen. Das System bekommt laufend die neuesten Pakete.

Block 6

Ü6.1 Das geht nicht. Hyper-V braucht Windows 11 Pro, Enterprise oder Education. Alternative laut Folien: VirtualBox. Außerdem muss die Hardware-Virtualisierung aktiviert sein.

Ü6.2

- `sudo`: mit Administratorrechten ausführen.
- `apt update`: Paketlisten aktualisieren.
- `&&`: zweiten Befehl nur bei Erfolg des ersten ausführen.
- `apt upgrade`: neuere Versionen der installierten Pakete installieren.
- `-y`: alle Bestätigungsfragen automatisch mit “Ja” beantworten.

Ü6.3 Nein. `&&` startet den zweiten Befehl nur, wenn der erste erfolgreich war. Schlägt `apt update` fehl, läuft `apt upgrade` nicht.

📋 Schlagwortliste für die Prüfung

Begriff	Kurzdefinition	Block
Betriebssystem	Software, die Anwendungen ausführt und Systemressourcen verwaltet	2
Kernel	Kern des Betriebssystems, Vermittler zwischen Anwendungen und Hardware	2
Userspace	Bereich, in dem die normalen Programme laufen, getrennt vom Kernel	2
Schichtenmodell	Anwender → Anwendungssoftware → Betriebssystem → Hardware	2
unixoid	ähnlich wie Unix aufgebaut, ohne Unix-Code	2
Unix-Philosophie	kleine Programme, die eine Sache gut tun	2
Alles ist eine Datei	Geräte, Prozesse, Netzwerke erscheinen als Dateien	2
Multiuser-System	mehrere Benutzer arbeiten gleichzeitig	2
root	Administrator-Konto mit allen Rechten	2
Unix	altes Betriebssystem, Ende 1960er, Bell Labs (AT&T), Thompson und Ritchie	3
BSD	Berkeley Software Distribution, 1977, freie Unix-Abspaltung	3
FreeBSD / OpenBSD / NetBSD	freie Nachfolger von BSD	3
MINIX	einfaches Lehr-Betriebssystem, von Torvalds genutzt	3
Linus Torvalds	begann 1991 in Helsinki den Linux-Kernel	3
Tux	Linux-Pinguin, 1996 von Larry Ewing mit GIMP gezeichnet	3
GPL	GNU General Public License, Lizenz von Linux seit 1992	3, 4
GNU	“GNU’s Not Unix”, rekursives Akronym, 1983 von Richard Stallman	4
GNU/Linux	GNU-Werkzeuge + Linux-Kernel	4

Begriff	Kurzdefinition	Block
GCC / glibc / Bash / Coreutils	Compiler / C-Bibliothek / Shell / Grundbefehle	4
Open Source	Quellcode offen, darf eingesehen, verändert, weitergegeben werden	4
Freie Software	Freiheit zu verwenden, untersuchen, verändern, weitergeben	4
“free as in freedom”	frei = Freiheit, nicht kostenlos	4
Distribution	Kernel + Softwarepakete, optimiert für einen Einsatzzweck	5
LTS	Long Term Support, sehr lange Updates (Ubuntu LTS: bis 10 Jahre)	5
Rolling Release	laufende Aktualisierung ohne feste Versionen (Arch Linux)	5
Debian	stabil, konservativ, Basis vieler Distributionen, Codenamen aus Toy Story	5
Red Hat / RHEL	kommerzielle Enterprise-Distribution mit Support	5
YaST	zentrales Konfigurationswerkzeug von SUSE	5
Kali Linux	Distribution für Penetrationstests	5
Virtuelle Maschine	Computer im Computer, z. B. mit VirtualBox oder Hyper-V	6
Hyper-V	Virtualisierung von Windows, nicht in Windows 11 Home	6
Secure Boot	Vorlage “Microsoft UEFI-Zertifizierungsstelle” für Linux in Hyper-V	6
amd64	Architektur des Debian-ISO für normale 64-Bit-PCs	6
apt	Paketverwaltung von Debian	6
apt update / apt upgrade	Paketlisten laden / neuere Pakete installieren	6
apt install	ein Paket installieren	6
sudo	Befehl mit Administratorrechten ausführen	6

Begriff	Kurzdefinition	Block
`&&`	zweiter Befehl nur, wenn der erste erfolgreich war	6
`-y`	Bestätigungsfragen automatisch mit “Ja” beantworten	6

🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug

Grundlage: Folien “Linux, Modul 2: Grundlagen II” (GBS Fachschule). Die Zusammenfassung hat sechs Blöcke.

- 🎯 **Fett** gedruckte Begriffe sind Prüfungs-Schlagwörter. Du solltest sie erklären können.
- 📊 Jeder Abschnitt hat eine Zeile “IHK-Relevanz”. Das ist eine eigene Einschätzung. Es ist keine amtliche Angabe.
- 📝 Am Ende jedes Blocks stehen Übungsfragen. Die Lösungen findest du gesammelt am Ende.
- ➕ “Ergänzung” heißt: Das steht nicht auf den Folien. Es hilft aber beim Verstehen.

Die Symbole in diesem Dokument:

Symbol	Bedeutung
💡	Merksatz – das solltest du behalten
⚠️	Achtung – hier passieren oft Fehler
📊	IHK-Relevanz – wie wichtig ist das für die Prüfung?
🎓	Prüfungstipp
📝	Übungsfragen
🖼️	Bild zum Thema

Wo passt Linux in die IHK-Prüfung?

Der IHK-Leitfaden nennt keine Linux-Befehle. Linux passt aber zu drei Prüfungsteilen:

- **AP1** (Teil 1, “Einrichten eines IT-gestützten Arbeitsplatzes”): Betriebssysteme und IT-Sicherheit am Arbeitsplatz.
- **AP2 Konzeption und Administration**: Systeme verwalten. Aufgaben mit Skripten automatisieren.
- **AP2 Netzwerke**: Hier geht es vor allem um SSH. SSH ist ein sicheres Protokoll für die Fernwartung.

⌨️ **Prompt-Zeichen in den Beispielen:**

- `$` = Du arbeitest als normaler Benutzer.
- `#` = Du arbeitest als Root (Administrator).
- Das Zeichen gehört nicht zum Befehl. Du tippst es nicht mit.

🐚 Block 1: Die Shell

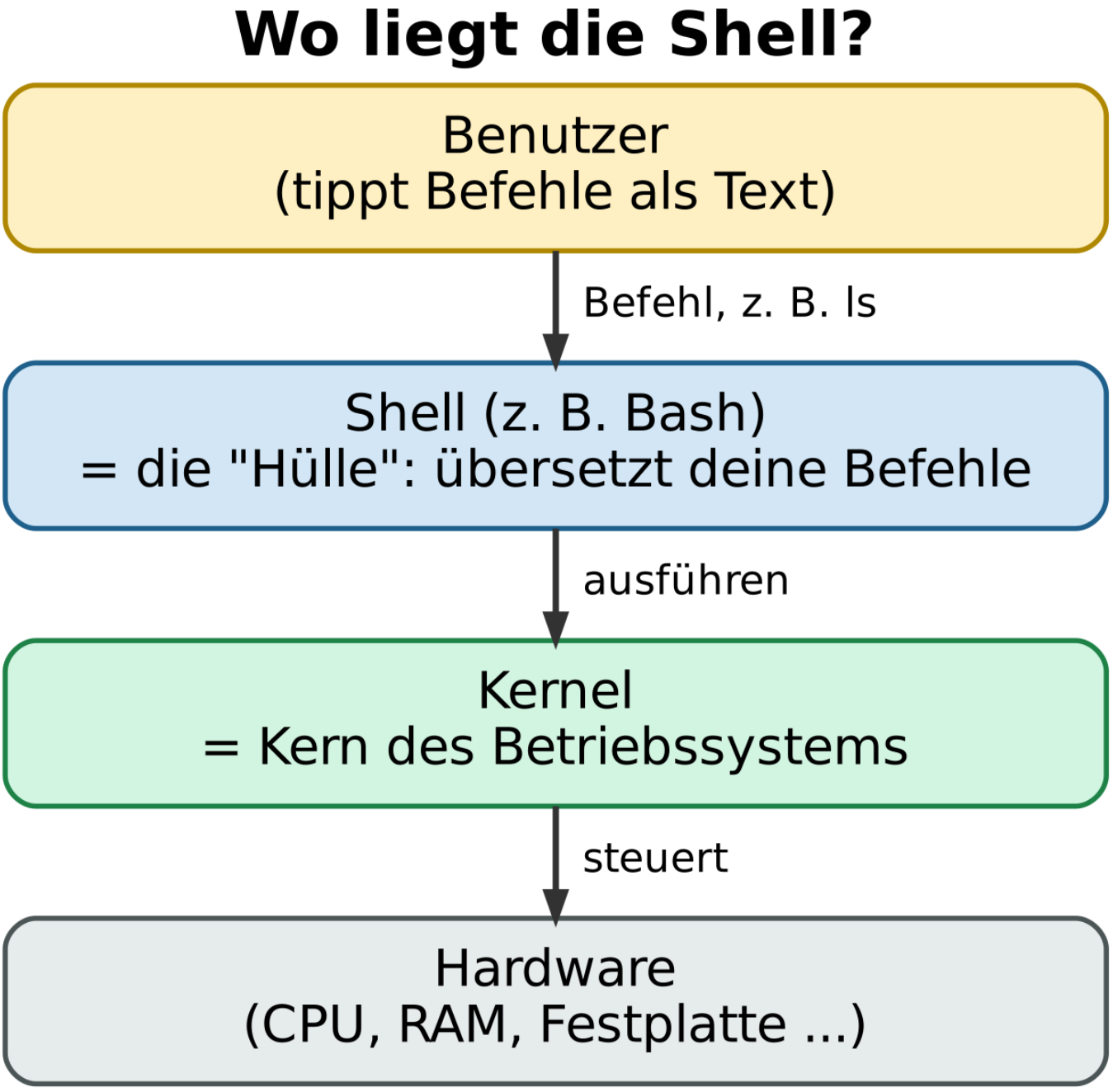
1.1 Was ist eine Shell? IHK mittel

> 🎯 **IHK: mittel** der Begriff Shell und ihre Rolle zwischen Benutzer und Kernel sind Grundwissen “Betriebssysteme” (AP1). Shell-Skripte sind Thema in AP2 Konzeption/Administration

Stell dir einen Dolmetscher vor. Du sprichst Deutsch. Dein Gegenüber versteht nur Japanisch. Der Dolmetscher übersetzt für dich.

Genau so arbeitet die **Shell**. Sie ist ein Programm. Sie nimmt deine Befehle entgegen. Sie versteht (interpretiert) sie. Dann führt sie die Befehle aus. Die Shell ist die Schnittstelle zwischen dir und dem Betriebssystem.

“Shell” heißt auf Deutsch “Hülle”. Die Shell liegt wie eine Schale um den Kern. Den Kern des Betriebssystems nennt man **Kernel**.



Schichtenbild: Benutzer, Shell, Kernel, Hardware

Schichtenbild: Benutzer, Shell, Kernel, Hardware

Eigenschaften der Shell:

- ⌨️ Du bedienst sie nur mit Text. Es gibt keine Maus und keine grafische Oberfläche.
- 🖥️ Die Antwort kommt auch als Text.
- 📜 Die Shell ist auch eine **Skriptsprache**. Du kannst Befehle in einer Datei sammeln. Die Shell führt sie dann automatisch aus.

  💡 **Merksatz:** Shell = Hülle um den Kernel. Sie übersetzt deine Befehle für das System.

1.2 Verschiedene Shells IHK gering

> 🎯 **IHK: gering** Namen und Herkunft der Shells werden kaum abgefragt. Wichtig ist nur: Bash ist der Standard

Es gibt mehrere Shells. Das ist wie bei Autos: Alle fahren, aber jedes hat Besonderheiten.

Shell	Name und Herkunft	Besonderheit
Bash	Bourne Again Shell; Wortspiel auf die Bourne Shell (`sh`) von Stephen Bourne	Standard-Shell unter Debian und vielen anderen Distributionen; Befehlshistorie, Tab-Vervollständigung
ksh	Korn Shell, David Korn, 1983, Bell Labs	Weiterentwicklung der Bourne Shell, starke Skriptfunktionen
zsh	Z Shell, 1990, Princeton; Name nach dem Login “zsh” des Assistenten Zhong Shao	sehr flexibel, viele Plugins und Themes, beliebt bei Entwicklern
tcsh	TENEX C Shell; Erweiterung der C Shell (`csh`)	Syntax teils an C angelehnt; heute selten

Die Bourne Shell (`sh`) stammt von Stephen Bourne. Sie war eine der ersten wichtigen Unix-Shells. Die Bash kann ihre Skripte ausführen.

ℹ️ Randnotiz: `neofetch` zeigt Systeminformationen an. Es wird seit 2024 nicht mehr weiterentwickelt. Der aktive Nachfolger heißt `fastfetch` .

1.3 Syntax-Schreibweise und Prompt IHK mittel

> 🎯 **IHK: mittel** die Schreibweise <befehl> [optionen] [argumente] braucht man, um Aufgaben und Manpages richtig zu lesen

Die Shell zeigt dir, wann sie bereit ist. Dafür steht am Zeilenanfang ein Zeichen. Dieses Zeichen heißt **Prompt** (Eingabeaufforderung).

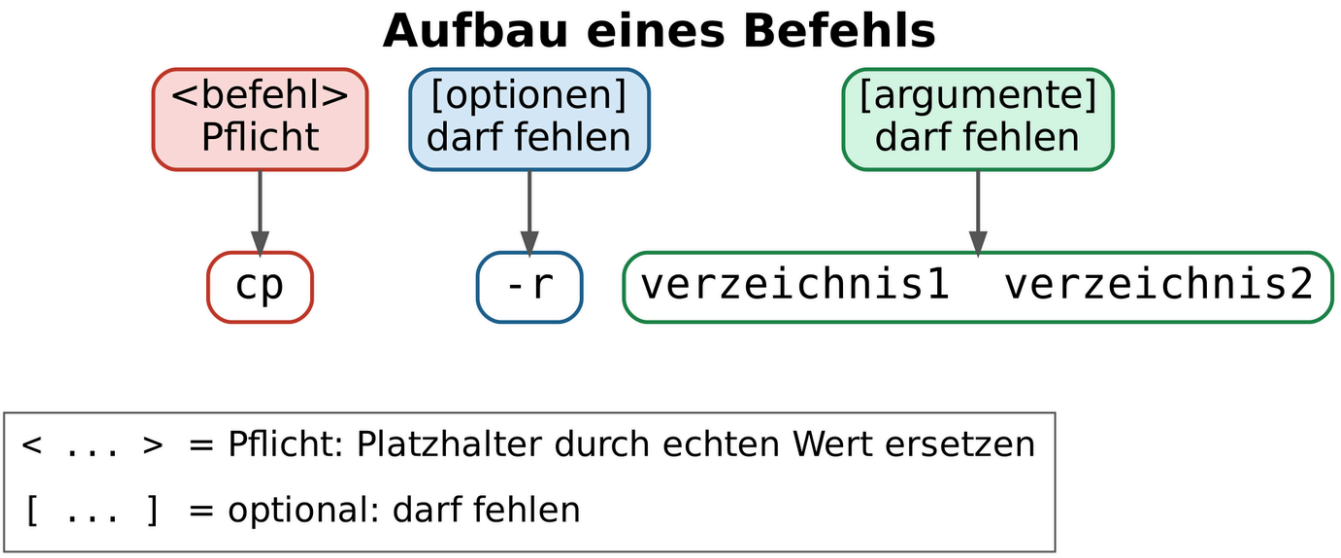
Zeichen	Bedeutung
`$`	normaler Benutzer
`#`	Root

Folien und Handbücher nutzen eine feste Schreibweise. Sie zeigt, was Pflicht ist und was nicht. Das ist wie bei einem Formular: Manche Felder musst du ausfüllen, andere nicht.

Schreibweise	Bedeutung	Beispiel
`[...]`	optional, darf fehlen	`ls [pfad]`
`<...>`	Pflicht, Platzhalter durch echten Wert ersetzen	`cat <datei>`

Allgemeine Befehlsform:

```
<befehl> [optionen] [argumente]
```



Aufbau eines Befehls am Beispiel cp -r

Aufbau eines Befehls am Beispiel cp -r

🔠 Linux achtet bei Dateinamen auf Groß- und Kleinschreibung. Der Fachbegriff dafür heißt **case-sensitive**.

- `Datei.txt`, `datei.txt` und `DATEI.TXT` sind drei verschiedene Dateien.
- Windows macht diesen Unterschied normalerweise nicht.

> ⚠️ **Achtung:** Ein falscher Großbuchstabe, und Linux findet die Datei nicht.

📝 Übungsfragen Block 1

Ü1.1 Erklären Sie den Begriff Shell. Wo liegt sie zwischen Benutzer und Hardware?

Ü1.2 In einer Anleitung steht `cp [optionen] <quelle> <ziel>`. Welche Teile müssen Sie angeben, welche dürfen fehlen?

Ü1.3 Ein Kollege legt `Bericht.txt` und `bericht.txt` im selben Linux-Verzeichnis an. Sind das zwei Dateien? Begründen Sie.

⌨️ Block 2: Grundlegende Befehle

2.1 Übersicht IHK mittel

> 🎯 **IHK: mittel** Grundbefehle werden selten einzeln abgefragt. Sie sind aber Voraussetzung für jede Shell- und Skriptaufgabe in AP2 Konzeption/Administration

Die Befehle sind meist Abkürzungen von englischen Wörtern. Wenn du das Wort kennst, merkst du dir den Befehl leichter.

Befehl	Herkunft	Zweck
`echo`	echo	Text ausgeben
`ls`	list	Verzeichnisinhalt auflisten
`cat`	concatenate	Dateiinhalt anzeigen, Dateien aneinanderhängen
`touch`	touch	leere Datei anlegen oder Zeitstempel aktualisieren
`cp`	copy	kopieren (Verzeichnisse mit `-r`)
`mv`	move	verschieben oder umbenennen
`rm`	remove	Dateien löschen, ohne Papierkorb
`mkdir`	make directory	Verzeichnis anlegen
`rmdir`	remove directory	**leeres** Verzeichnis löschen
`cd`	change directory	Verzeichnis wechseln

2.2 📣 echo und Umleitung IHK hoch

> 🎯 **IHK: hoch** Umleitung mit > und >> ist Grundlage für Skripte und Logdateien (AP2 Konzeption/Administration)

`echo` heißt “Echo”. Der Befehl gibt eine Zeile Text aus. In Skripten nutzt man ihn für Hinweise und zur Fehlersuche.

```
$ echo Hallo
Hallo
```

Normalerweise erscheint die Ausgabe auf dem Bildschirm. Du kannst sie aber in eine Datei “umlenken”. Das heißt **Umleitung**.

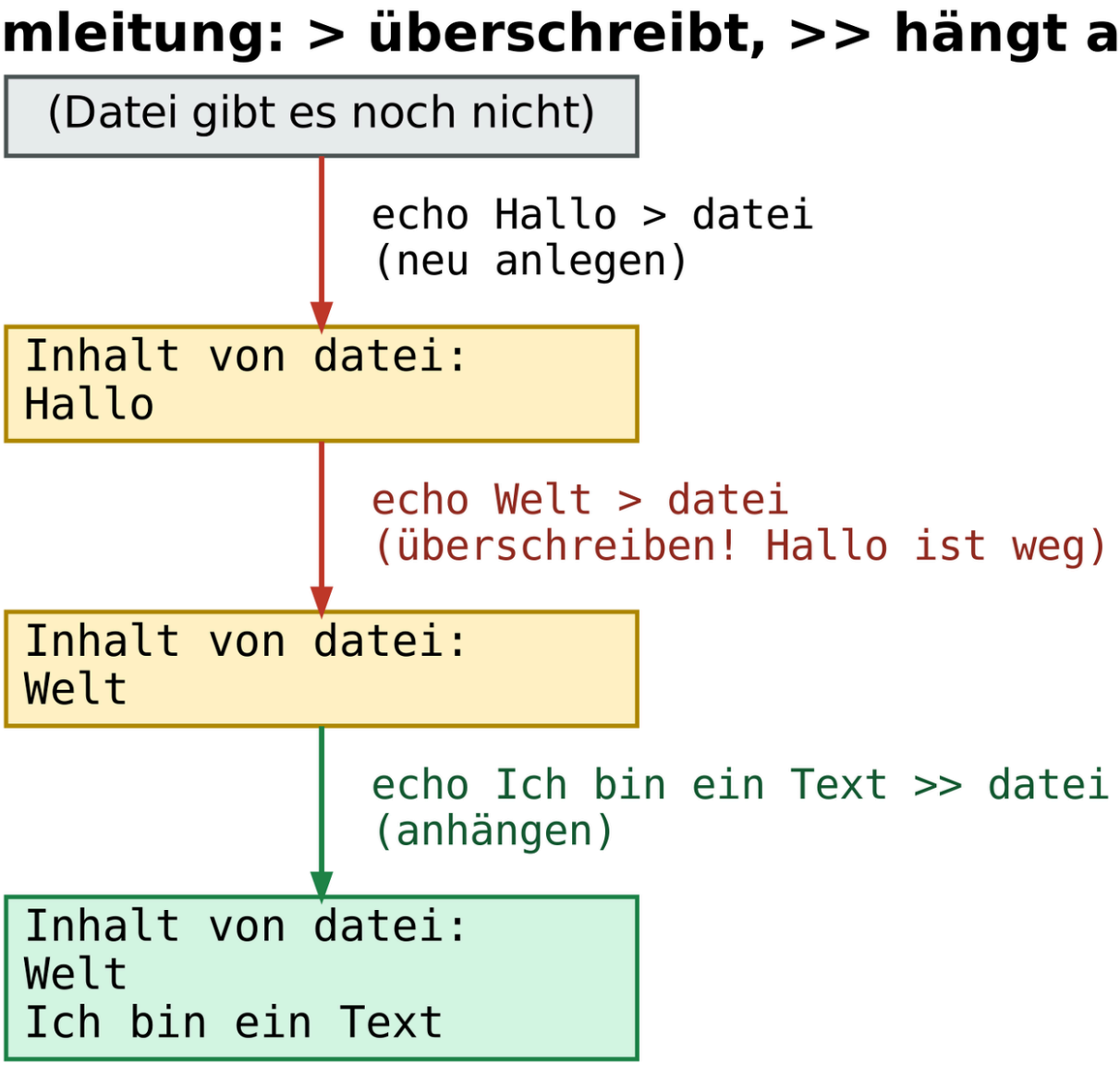
Vergleich aus dem Alltag:

- `>` ist wie ein Blatt **neu beschreiben**. Was vorher draufstand, ist weg.
- `>>` ist wie eine Zeile **unten dazuschreiben**. Das Alte bleibt stehen.

Zeichen	Wirkung
`>`	Datei wird **überschrieben** (oder neu angelegt)
`>>`	Ausgabe wird an das Dateiende **angehängt**

Beispiel 1: Überschreiben und Anhängen

```
$ echo Hallo > datei
$ cat datei
Hallo
$ echo Welt > datei
$ cat datei
Welt
$ echo Ich bin ein Text >> datei
$ cat datei
Welt
Ich bin ein Text
```



Vorher/Nachher: Inhalt der Datei bei > und >>

Vorher/Nachher: Inhalt der Datei bei > und >>

Erklärung Schritt für Schritt:

1. `>` legt `datei` an. Es schreibt “Hallo” hinein.
2. Das zweite `>` überschreibt den Inhalt. “Hallo” ist weg. Jetzt steht “Welt” darin.
3. `>>` hängt “Ich bin ein Text” als neue Zeile an.

⚠️ **Fallstrick:** Ein `>` zu viel löscht den alten Inhalt sofort. Es gibt keine Rückfrage.

2.3 📂 ls – Verzeichnisinhalt auflisten IHK mittel

> 🎯 **IHK: mittel** ls -l lesen ist Voraussetzung für Rechte-Aufgaben in AP2 Konzeption/Administration

`ls` zeigt dir, was in einem Ordner liegt. Das ist wie ein Blick in eine Schublade.

Syntax: `ls [optionen] [pfad]`

- Ohne Pfad zeigt `ls` den aktuellen Ordner.
- `ls` und `ls .` sind gleich.

Option	Bedeutung
`-l`	long format: Rechte, Besitzer, Größe, Datum
`-a`	all: auch versteckte Dateien (Name beginnt mit `.`)
`-la`	beides kombiniert

➕ Ergänzung zur Miniübung `ls -lisah`:

- `-i` zeigt die Inode-Nummer.
- `-s` zeigt den belegten Speicherplatz in Blöcken.
- `-h` macht Größen lesbar, z. B. `4,0K`.
- Nachschlagen kannst du das mit `man ls`.

Beispiele:

Befehl	Zeigt
`ls /etc`	Konfigurationsdateien des Systems
`ls /usr/bin`	ausführbare Programme
`ls /proc`	laufende Prozesse und Systeminformationen

> 💡 **Unix-Philosophie:** Ein kleines Programm soll eine Sache gut machen. `ls` ist dafür ein Beispiel.

2.4 📄 cat – Dateiinhalt anzeigen IHK mittel

> 🎯 **IHK: mittel** häufig in Befehlsketten und beim Prüfen von Konfigurationsdateien

`cat` zeigt den Inhalt einer Datei auf dem Bildschirm.

Syntax: `cat <datei>`

Befehl	Zeigt
`cat /proc/cpuinfo`	Informationen zur CPU
`cat /proc/meminfo`	Informationen zu RAM und Swap
`cat /etc/debian_version`	installierte Debian-Version
`cat /etc/passwd`	Benutzerkonten

“cat” kommt von **concatenate**. Das heißt “aneinanderhängen”. So fügst du zwei Dateien zu einer zusammen:

```
$ cat datei1.txt datei2.txt > zusammen.txt
```

💡 Für lange Dateien ist `less` besser. Es zeigt den Text Seite für Seite (siehe Block 4).

2.5 👆 touch – Datei anlegen oder Zeitstempel setzen IHK gering

> 🎯 **IHK: gering** nützliches Werkzeug, kaum eigene Prüfungsfrage

`touch` heißt “berühren”. Der Befehl macht zwei Dinge:

- Gibt es die Datei noch nicht, legt er eine **leere** Datei an.
- Gibt es die Datei schon, setzt er die Zeitstempel neu. Zeitstempel sind: letzter Zugriff und letzte Änderung. Der Inhalt bleibt gleich.

```
$ touch hello world foo bar
```

Dieser Befehl legt vier leere Dateien auf einmal an.

Wozu Zeitstempel ändern? Die Folien lassen das offen. Typische Fälle:

- 💾 Eine Datei soll beim nächsten zeitgesteuerten Backup als “geändert” gelten.
- 🔧 Build-Werkzeuge wie `make` übersetzen nur Dateien, die neuer sind als das Ergebnis. `touch` erzwingt eine Neuübersetzung.
- 🔍 Man testet Skripte, die nach Datum suchen, z. B. `find -mtime`.
- 🧪 Man legt schnell Testdateien an.

2.6 📋 cp, mv, rm – kopieren, verschieben, löschen IHK mittel

> 🎯 **IHK: mittel** Dateioperationen gehören zu jeder Administrationsaufgabe. rm ohne Papierkorb ist ein typischer Sicherheitshinweis

Vergleich mit Papier:

- `cp` ist der **Kopierer**. Das Original bleibt, eine Kopie kommt dazu.
- `mv` ist **Umräumen** oder ein **neues Etikett**. Es entsteht keine Kopie.
- `rm` ist der **Schredder**. Weg ist weg.

Befehl	Syntax	Wirkung
`cp`	`cp <quelle> <ziel>`	Kopie anlegen, Original bleibt
`cp -r`	`cp -r <verz> <ziel>`	Verzeichnis samt Inhalt kopieren (rekursiv)

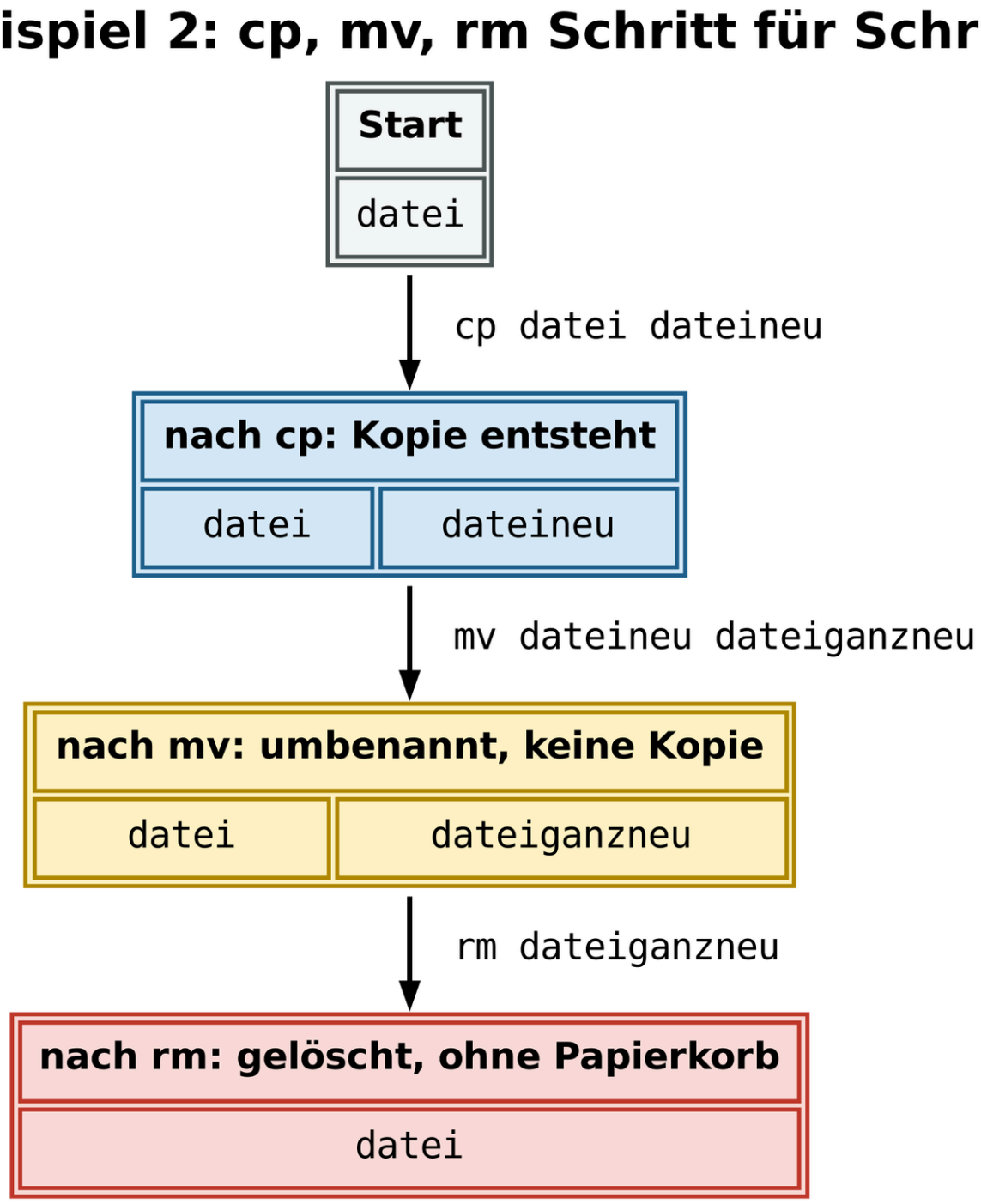
Befehl	Syntax	Wirkung
`mv`	`mv <quelle> <ziel>`	verschieben, es entsteht keine Kopie
`mv`	`mv alt.txt neu.txt`	umbenennen
`rm`	`rm <datei>`	Datei löschen
`rm -r`	`rm -r <verz>`	Verzeichnis samt Inhalt löschen

> ⚠️ **Achtung:** `rm` löscht sofort und **ohne Papierkorb**. Bei `rm -r` ist der ganze Verzeichnisbaum weg. Dann hilft nur noch ein Backup.

Beispiel 2: Quiz cp, mv, rm

Ausgangslage: Im Verzeichnis liegt nur `datei`.

```
$ cp datei dateineu
$ ls
datei  dateineu
$ mv dateineu dateiganzneu
$ ls
datei  dateiganzneu
$ rm dateiganzneu
$ ls
datei
```



Ablauf von Beispiel 2: Dateien nach cp, mv und rm

Ablauf von Beispiel 2: Dateien nach cp, mv und rm

1. `cp` legt eine Kopie an. Sie heißt `dateineu`.
2. `mv` benennt `dateineu` in `dateiganzneu` um. Es bleiben zwei Dateien.
3. `rm` löscht `dateiganzneu`. Übrig bleibt nur das Original.

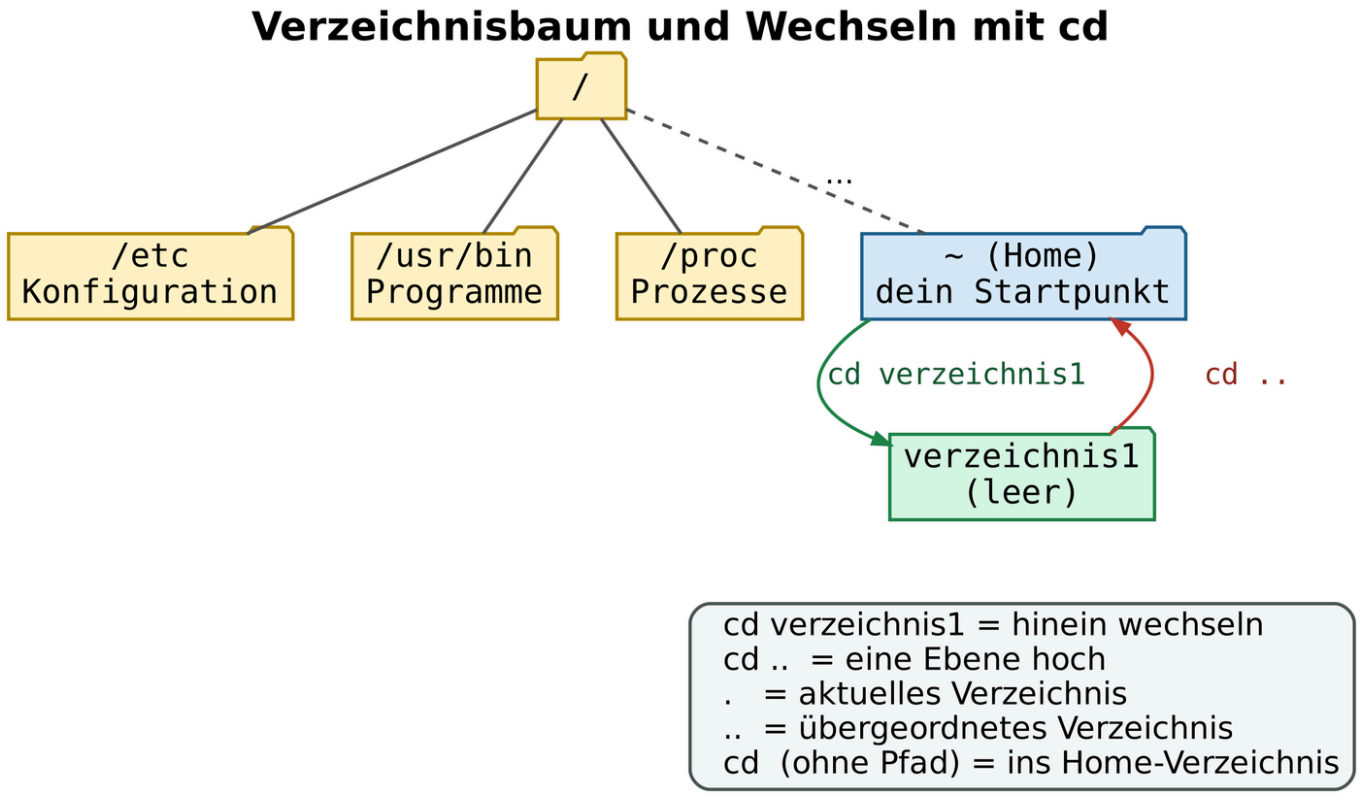
2.7 🗂️ mkdir, rmdir, cd – mit Verzeichnissen arbeiten IHK mittel

> 🎯 **IHK: mittel** Grundwissen für Pfadaufgaben und Skripte

Ein Verzeichnis ist ein Ordner. Die Ordner bilden einen Baum. Ganz oben steht die Wurzel `/`.

Befehl	Wirkung
`mkdir ordner`	Verzeichnis anlegen
`mkdir ordner1 ordner2 ordner3`	mehrere Verzeichnisse auf einmal
`rmdir ordner`	Verzeichnis löschen, **nur wenn leer**
`rm -r ordner`	Verzeichnis samt Inhalt löschen
`cd ordner`	in `ordner` wechseln
`cd ..`	eine Ebene nach oben
`cd .`	im aktuellen Verzeichnis bleiben (gültig, aber nutzlos)
`cd`	ohne Pfad: ins eigene Home-Verzeichnis

💡 **Merksatz:** Ein Punkt `.` = hier. Zwei Punkte `..` = eine Ebene höher.



Verzeichnisbaum und Wechseln mit cd

Verzeichnisbaum und Wechseln mit cd

Beispiel 3: Quiz mkdir und cd

Ausgangslage: ein leeres Verzeichnis.

```
$ mkdir verzeichnis1
$ ls
verzeichnis1
$ cd verzeichnis1
$ ls
$ cd ..
$ ls
verzeichnis1
```

1. `mkdir` legt `verzeichnis1` an. `ls` zeigt es.
2. Mit `cd verzeichnis1` gehst du hinein. `ls` zeigt nichts. Der Ordner ist leer.
3. `cd ..` führt eine Ebene zurück. `ls` zeigt wieder `verzeichnis1`.

War der Ausgangsordner nicht leer? Dann zeigt `ls` auch die anderen Einträge.

📝 Übungsfragen Block 2

Ü2.1 Die Datei `log.txt` enthält drei Zeilen. Sie führen `echo Neustart > log.txt` aus. Was steht danach in der Datei? Wie hätten Sie die Zeile anhängen müssen?

Ü2.2 `rmdir projekt` meldet einen Fehler. Nennen Sie die wahrscheinliche Ursache und den Befehl, der das Verzeichnis trotzdem löscht. Worauf müssen Sie achten?

Ü2.3 Nennen Sie zwei sinnvolle Einsatzzwecke für `touch`.

⚙️ Block 3: Befehlsoptionen und Hilfe

3.1 🎛️ Optionen IHK mittel

> 🎯 **IHK: mittel** Optionen richtig lesen und schreiben ist Grundlage für Skriptaufgaben in AP2 Konzeption/Administration

Optionen sind wie Extras bei einer Bestellung. “Pizza” ist der Befehl. “Mit extra Käse” ist die Option. Man nennt sie **Optionen**. Sie ändern oder erweitern, was ein Befehl tut.

Art	Schreibweise	Beispiel
Kurzoption	ein Bindestrich, ein Buchstabe	`-l`, `-a`
kombinierte Kurzoptionen	ein Bindestrich, mehrere Buchstaben	`-la`

Art	Schreibweise	Beispiel
Langoption	zwei Bindestriche, ganzes Wort	`--verbose`, `--recursive`

Beispiel mit `cp`:

- `-r` (recursive) kopiert Verzeichnisse mit Inhalt.
- `-v` (verbose = gesprächig) zeigt jede kopierte Datei an.

Diese drei Befehle machen genau dasselbe:

```
$ cp -r -v verzeichnis1 verzeichnis2
$ cp -rv verzeichnis1 verzeichnis2
$ cp --recursive --verbose verzeichnis1 verzeichnis2
```

> 🎓 **Tipp:** In Skripten sind Langoptionen oft besser. Man kann sie leichter lesen.

3.2 🆘 Hilfe unter Linux IHK mittel

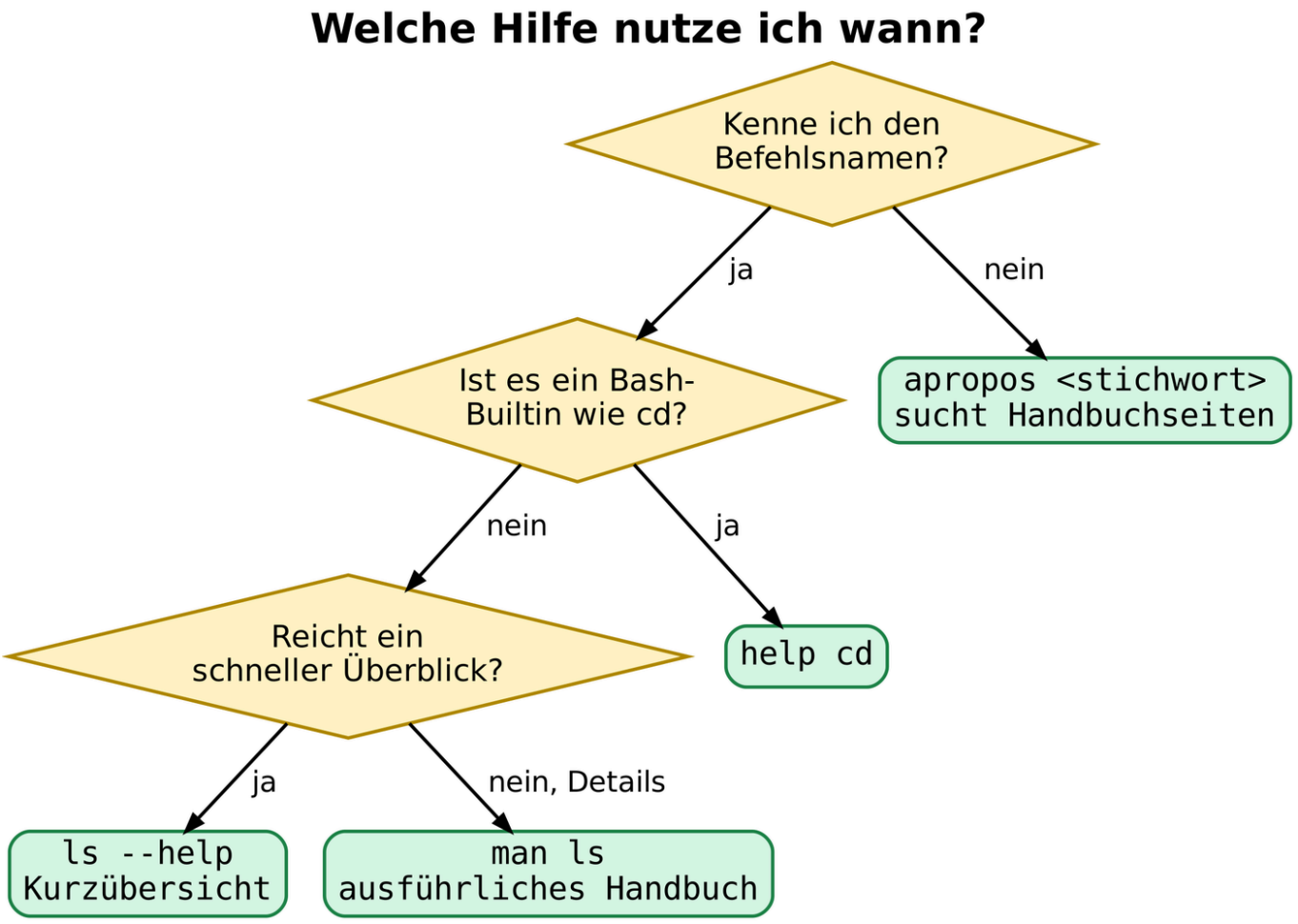
> 🎯 **IHK: mittel** “Wie helfe ich mir selbst?” ist eine typische Praxisfrage. In AP1 passt sie zu Betriebssystemen und Kundeneinweisung

Linux hat mehrere Hilfen. Das ist wie bei einem Gerät: Es gibt den Kurzaufkleber, die dicke Anleitung und das Stichwortverzeichnis.

Form	Wofür	Beispiel
`help`	in die Bash **eingebaute** Befehle (Builtins), z. B. `cd`, `history`, `export`	`help history`
`--help`	kurze Übersicht zu Syntax und Optionen	`ls --help`
`man`	ausführliches Handbuch (manual)	`man ls`
`apropos`	sucht Handbuchseiten zu einem Stichwort	`apropos kernel`

Wann nehme ich was?

- Befehl bekannt, schneller Überblick: `--help`.
- Befehl bekannt, Details nötig: `man`.
- Befehl unbekannt: `apropos <stichwort>`.
- Befehl ist ein Bash-Builtin wie `cd`: `help`.



Entscheidungsbaum: welche Hilfe wann

Entscheidungsbaum: welche Hilfe wann

🔍 So bedienst du man

Taste	Wirkung
`q`	beenden
Pfeiltasten, Bild hoch/runter	blättern
`/begriff` + Enter	vorwärts suchen
`?begriff` + Enter	rückwärts suchen
`n` / `N`	nächster / vorheriger Treffer

> 💡 **Merksatz der Folien:** Ein guter Administrator kennt nicht jede Option auswendig. Er weiß, wie er sich selbst hilft.

📝 Übungsfragen Block 3

Ü3.1 Schreiben Sie `ls -l -a -h` auf zwei andere Arten.

Ü3.2 Sie suchen einen Befehl zum Thema “partition”, kennen aber den Namen nicht. Welches Hilfswerkzeug nutzen Sie?

Ü3.3 `man cd` liefert auf einem Debian-System keine Seite. Warum, und wie erhalten Sie Hilfe zu `cd`?

🛠️ Block 4: Arbeiten mit der Bash

4.1 ⏪ Historie, Suche, Vervollständigung IHK gering

> 🎯 **IHK: gering** Komfortfunktionen für die Praxis; selten Prüfungsstoff

Die Bash merkt sich deine Befehle. Das ist wie der Verlauf im Browser. Diese Liste heißt **Befehlshistorie**.

Funktion	Taste / Befehl	Wirkung
Befehlshistorie	Pfeil hoch / runter	frühere Befehle durchblättern
	`history`	alle gespeicherten Befehle ausgeben
Rückwärtssuche	Strg + R	in der Historie suchen; erneut Strg + R für weitere Treffer
	Enter / Pfeil rechts	Treffer ausführen / zum Bearbeiten übernehmen
Tab-Vervollständigung	Tab	Befehl oder Pfad ergänzen, wenn eindeutig
	Tab Tab	alle Möglichkeiten anzeigen

💡 Die Tab-Taste ist wie die Wortvorschläge am Handy. Du tippst den Anfang, die Bash ergänzt den Rest.

Scrollen im GNOME-Terminal:

Tasten	Wirkung
Shift + Bild hoch/runter	seitenweise
Shift + Strg + Pfeil hoch/runter	zeilenweise
Shift + Pos1 / Ende	ganz nach oben / unten

🖥️ Terminal und Shell sind zwei Programme:

- Das **GNOME-Terminal** stellt das Fenster bereit. Es ist wie der Bilderrahmen.
- Die **Bash** läuft darin und führt die Befehle aus. Sie ist das Bild im Rahmen.

Quizfragen der Folien (Lösung aus Allgemeinwissen):

- GNOME stand ursprünglich für “GNU Network Object Model Environment”.
- GNU steht für “GNU’s Not Unix”. Das ist ein rekursives Akronym. Die Abkürzung steckt in sich selbst.

4.2 📜 less IHK gering

> 🎯 **IHK: gering** Werkzeug für die Praxis, z. B. beim Lesen von Logdateien

`less` zeigt lange Textdateien Seite für Seite. Du kannst blättern und suchen. Die Tasten sind wie bei `man`: `q` beendet, `/` sucht.

```
$ less /var/log/syslog
```

> 💡 **Faustregel:** kurze Datei mit `cat`, lange Datei mit `less`.

4.3 🏷️ alias IHK gering bis mittel

> 🎯 **IHK: gering bis mittel** Aliase zeigen den Umgang mit ~/.bashrc . Grundlage für die Anpassung der Arbeitsumgebung

Ein Spitzname ist kürzer als der volle Name. Genau so funktioniert ein **Alias**. Er ist ein selbst gemachter Kurzbefehl für einen langen Befehl.

Syntax:

```
$ alias kurzname='langer befehl'
```

Beispiel:

```
$ alias ll='ls -l --color=auto'
$ ll
```

> ⚠️ **Achtung:** Vor und nach dem `=` steht **kein** Leerzeichen.

> ⚠️ Ein Alias aus dem Terminal gilt nur, bis du das Fenster schließt.

💾 Alias dauerhaft speichern

1. Datei öffnen: `nano ~/.bashrc`
2. Zeile am Ende eintragen: `alias ll='ls -l --color=auto'`

3. Speichern mit Strg + O, Enter. Beenden mit Strg + X.
4. Neu laden: `source ~/.bashrc` oder ein neues Terminal öffnen.

Ein Alias darf auch wie ein vorhandener Befehl heißen, z. B. `ls`. Dann gilt der Alias vor dem Originalbefehl.

➕ Ergänzung: `alias` ohne Argument zeigt alle aktiven Aliase. `unalias ll` entfernt einen Alias in der laufenden Sitzung.

📝 Übungsfragen Block 4

Ü4.1 Sie haben vor einer Stunde einen langen `find`-Befehl eingegeben. Wie finden Sie ihn am schnellsten wieder?

Ü4.2 Legen Sie einen Alias `la` für `ls -la` an, der auch nach einem Neustart gilt. Beschreiben Sie die Schritte.

✏️ Block 5: Texteditoren

5.1 Überblick IHK gering

> 🎯 **IHK: gering** Editoren selbst werden kaum geprüft. Man braucht sie aber für jede Konfigurationsdatei und jedes Skript

Ein Texteditor ist wie ein einfaches Notizbuch. Er bearbeitet reine Textdateien. Beispiele sind Konfigurationsdateien und Skripte.

Auf einem Server gibt es oft keine grafische Oberfläche. Dann brauchst du einen Editor im Terminal.

Editor	Vorteil	Nachteil
vi	Klassiker, auf fast jedem System vorhanden	ungewohnte Bedienung
vim (vi improved)	vi-kompatibel, Syntax-Hervorhebung, Undo-Historie, Plugins	nicht immer vorinstalliert, Übung nötig
nano	sehr einfach, ideal für Einsteiger	weniger mächtig
emacs	sehr mächtig	sehr komplex
JOE (Joe’s Own Editor)	einfach, alle Alltagsfunktionen	meist nicht vorinstalliert

Auf einem typischen Debian sind vi und nano schon installiert. nano ist eine freie Alternative zu pico. Man bedient beide fast gleich.

5.2 💾 Speichern und Beenden IHK gering

> 🎯 **IHK: gering** Praxiswissen. Ein Prüfer erwartet eher, dass man einen Editor sicher bedienen kann

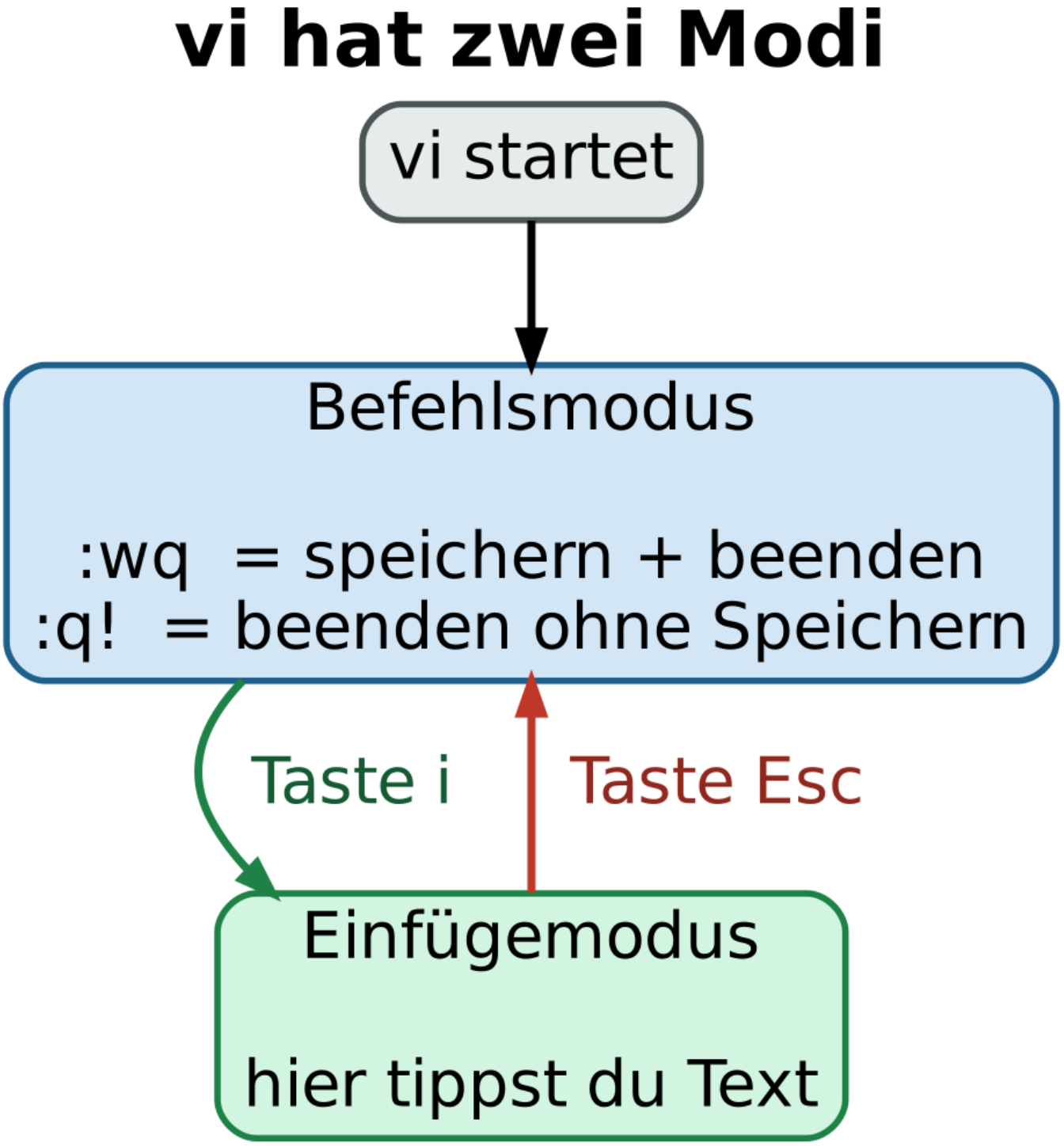
Editor	Speichern und beenden	Ohne Speichern beenden
vi / vim	Esc, dann `:wq`, Enter	Esc, dann `:q!`, Enter
nano	Strg + O, Enter, dann Strg + X	Strg + X, dann N
emacs	Strg + X, Strg + S; dann Strg + X, Strg + C	Strg + X, Strg + C; Speichern mit `n` ablehnen, mit `yes` bestätigen
JOE	Strg + K, dann X	Strg + C, dann `y`

🔄 vi verstehen: zwei Modi

Stell dir einen Kugelschreiber mit Mine vor. Mine drin: Du kannst schreiben. Mine draußen: Du kannst nur zeigen und Befehle geben.

vi hat zwei solche Zustände. Man nennt sie **Befehlsmodus** und **Einfügemodus**.

- Nach dem Start bist du im Befehlsmodus.
- Mit `i` wechselst du in den Einfügemodus. Jetzt kannst du schreiben.
- Mit Esc gehst du zurück in den Befehlsmodus.
- Erst dann funktionieren Befehle wie `:wq`.



Die zwei Modi von vi und der Wechsel mit i und Esc

Die zwei Modi von vi und der Wechsel mit i und Esc

Die Buchstaben bedeuten:

- `w` = write (speichern)
- `q` = quit (beenden)
- `!` = erzwingen

Zum Üben:

```
$ sudo apt install vim
$ vimtutor
```

`vimtutor` ist ein Lernprogramm zum Mitmachen. Es dauert etwa 30 Minuten. nano zeigt die wichtigsten Tasten unten im Fenster. Strg + G öffnet die ausführliche Hilfe.

📝 Übungsfragen Block 5

Ü5.1 Sie haben in vi versehentlich eine Konfigurationsdatei verändert. Wie beenden Sie vi, ohne zu speichern?

Ü5.2 Warum sollte ein Administrator wenigstens die Grundbedienung von vi beherrschen, obwohl nano einfacher ist?

🔐 Block 6: SSH (Secure Shell)

6.1 Was ist SSH? IHK hoch

> 🎯 **IHK: hoch** SSH ist das Standardprotokoll für sichere Fernwartung. Port 22 und der Telnet-Vergleich passen zu AP1 und AP2 Netzwerke

Vergleich: Eine Postkarte kann jeder Postbote lesen. Einen versiegelten Brief nicht.

- **Telnet** ist die Postkarte. Alles geht im Klartext durchs Netz, auch Passwörter.
- **SSH** ist der versiegelte Brief. Passwörter, Befehle und Ausgaben sind verschlüsselt.

SSH (Secure Shell) ist ein Protokoll. Damit verwaltest du einen Rechner sicher aus der Ferne. Ein **SSH-Client** baut eine **verschlüsselte** Verbindung zu einem **SSH-Server** auf. SSH ersetzt ältere, unsichere Verfahren wie **Telnet**.



Telnet überträgt im Klartext, SSH verschlüsselt

Telnet überträgt im Klartext, SSH verschlüsselt

Merkmal	Telnet	SSH
Verschlüsselung	nein, Klartext	ja
Standard-Port	TCP 23 (Ergänzung)	**TCP 22**
Einsatz heute	nicht mehr empfohlen	Standard

> 🎓 **Prüfungstipp:** SSH = verschlüsselt = **TCP 22**. Telnet = Klartext.

Was SSH kann:

- eine Shell auf dem entfernten Rechner öffnen, in der du arbeitest,
- einzelne Befehle entfernt ausführen,
- Dateien sicher übertragen. ➕ Ergänzung: Dafür gibt es die Werkzeuge `scp` und `sftp`.

6.2 Voraussetzungen und Syntax IHK hoch

> 🎯 **IHK: hoch** Voraussetzungen einer Verbindung abprüfen ist eine typische Fehlersuche-Aufgabe

✅ **Voraussetzungen** – alle drei müssen stimmen:

1. 🌐 Client und Server haben eine funktionierende Netzwerkverbindung.
2. 🖥️ Auf dem Zielsystem läuft ein SSH-Server. Er nimmt Verbindungen an.
3. 🚪 Der SSH-Port (Standard TCP 22) ist erreichbar. Keine Firewall sperrt ihn.

Syntax:

```
ssh [optionen] [benutzername@]server
```

Beispiel	Bedeutung
`ssh stephan@192.168.117.157`	als stephan mit dem Rechner per IP verbinden
`ssh 192.168.117.157`	lokaler Benutzername wird übernommen
`ssh stephan@server01.schule.local`	Verbindung über Hostnamen

Option	Bedeutung
`-p <port>`	anderer Port als 22
`-i <datei>`	bestimmten privaten Schlüssel nutzen (identity file)
`-v`, `-vv`, `-vvv`	ausführliche Meldungen zur Fehlersuche; mehr v = mehr Details

👀 **Woran erkennst du die Fernsitzung?** Der Prompt ändert sich. Beispiel: Aus `stephan@vmhost2` wird `stephan@pxesrv`. Alle Befehle laufen jetzt auf dem entfernten Rechner.

SSH-Clients:

System	Client
Linux	meist vorhanden, sonst per Paketverwaltung
macOS	im Terminal vorhanden
Windows	OpenSSH-Client in PowerShell, Eingabeaufforderung oder Windows Terminal; Komfort-Tools: PuTTY, MobaXterm

6.3 🔧 SSH-Server einrichten IHK mittel

> 🎯 **IHK: mittel** Dienste installieren, starten und aktivieren ist Administrationspraxis (AP2 Konzeption/Administration)

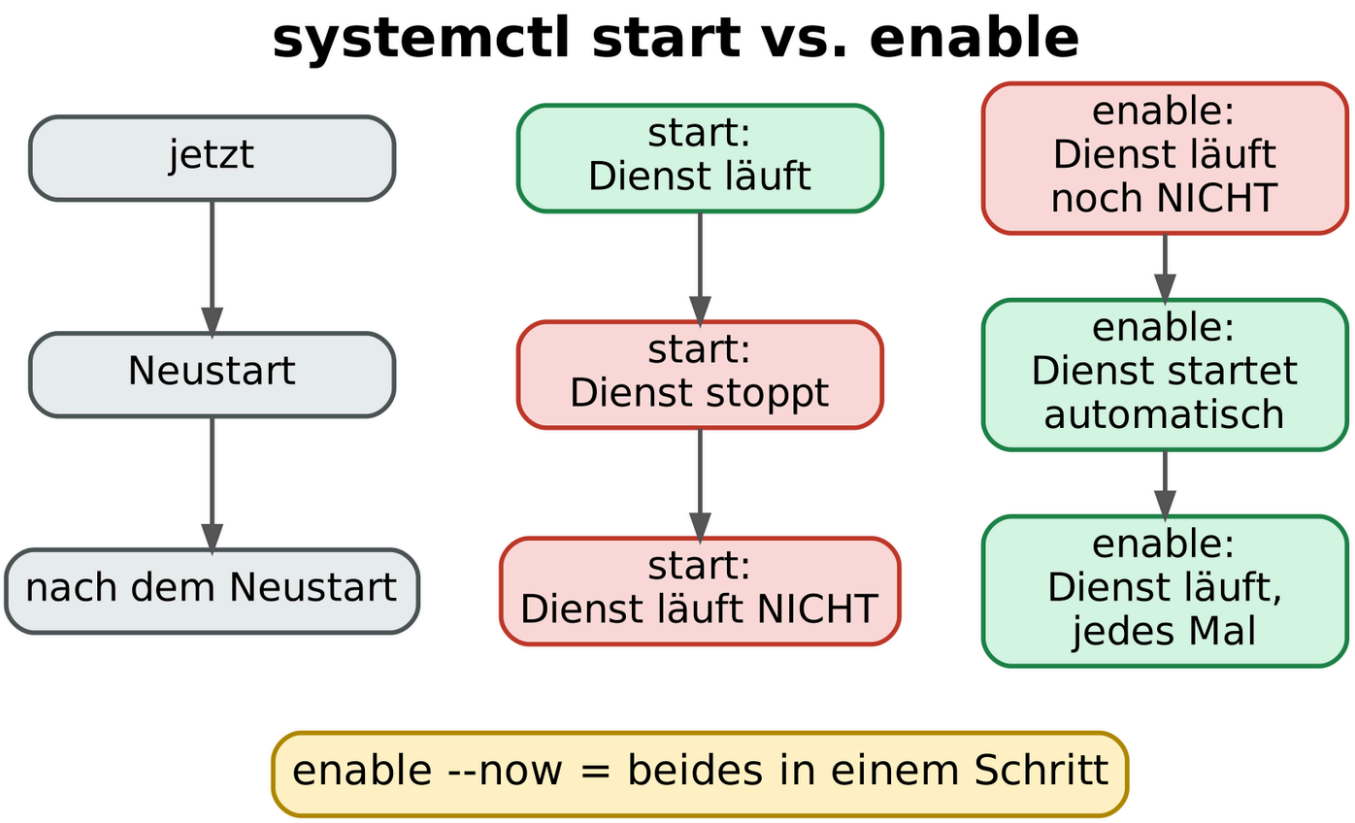
Auf dem Zielsystem (als Root):

```
# apt update
# apt install openssh-server
# systemctl start ssh
# systemctl enable ssh
# systemctl status ssh
```

Befehl	Wirkung
`apt update`	Paketlisten aktualisieren
`apt install openssh-server`	SSH-Server installieren
`systemctl start ssh`	Dienst **jetzt** starten
`systemctl enable ssh`	Dienst **bei jedem Systemstart** automatisch starten
`systemctl status ssh`	prüfen, ob der Dienst läuft

Unterschied `start` **und** `enable` – ein Vergleich mit Licht:

- `start` ist wie Licht anschalten. Es brennt sofort. Nach einem Stromausfall (Neustart) ist es aus.
- `enable` ist wie eine Zeitschaltuhr. Sie schaltet nicht sofort. Aber ab dem nächsten Start schaltet sie jedes Mal ein.



Zeitachse: was start und enable bewirken

Zeitachse: was start und enable bewirken

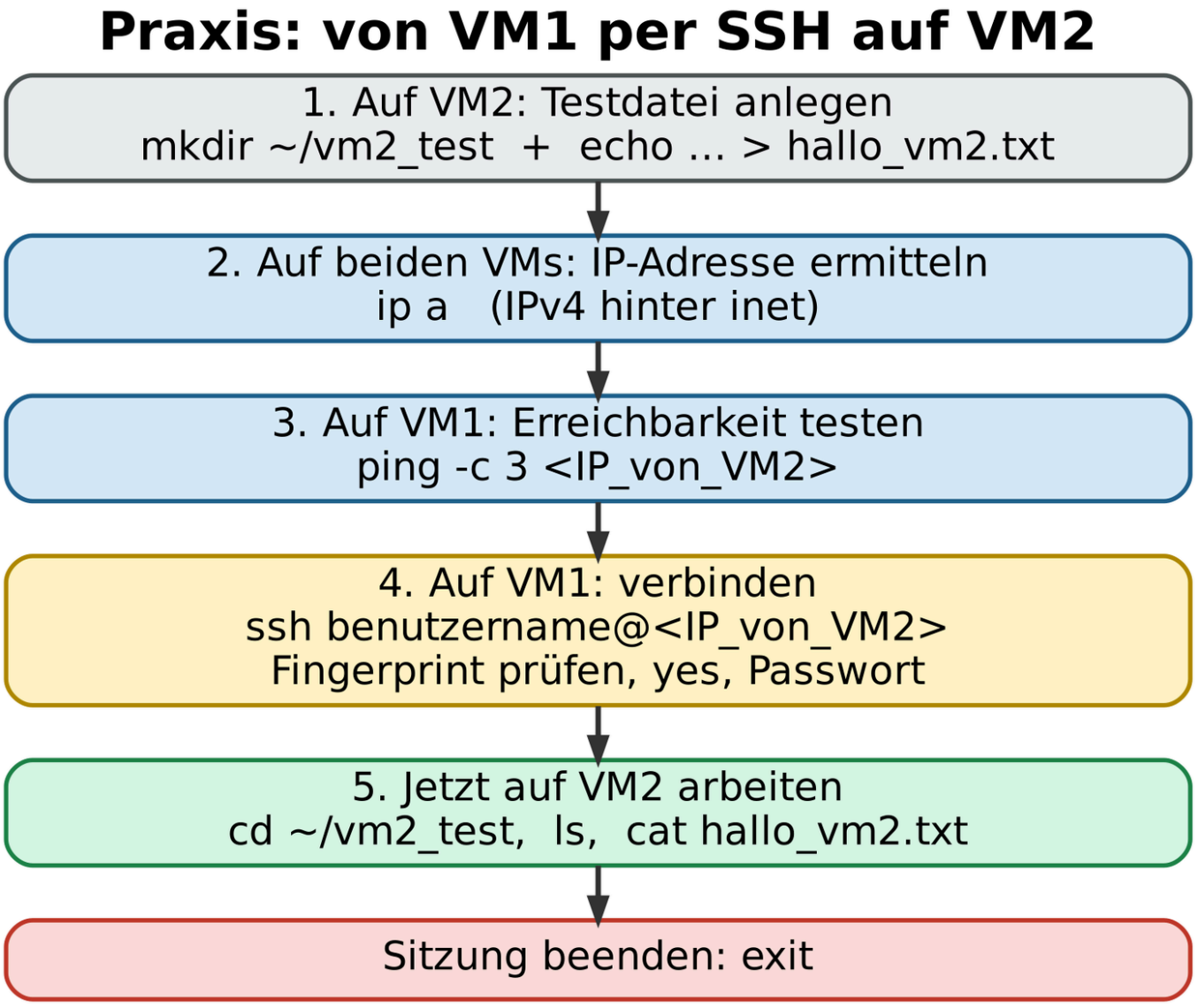
> 💡 **Merksatz:** `start` wirkt sofort, aber nur bis zum Neustart. `enable` wirkt erst ab dem nächsten Start, dann aber dauerhaft.

➕ Ergänzung: `systemctl enable --now ssh` macht beides in einem Schritt.

6.4 🖧 Praxis: Von VM1 per SSH auf VM2 IHK mittel

> 🎯 **IHK: mittel** der Ablauf (IP ermitteln, Erreichbarkeit testen, verbinden, Fingerprint prüfen) zeigt systematische Fehlersuche

🎯 Ziel: Du sitzt an VM1. Von dort findest du eine Datei auf VM2 und zeigst sie an.



Ablauf der SSH-Übung in fünf Schritten

Ablauf der SSH-Übung in fünf Schritten

Schritt 1: Testdatei auf VM2 anlegen

```
$ mkdir ~/vm2_test
$ echo "Ich bin eine Textdatei auf VM2" > ~/vm2_test/hallo_vm2.txt
```

Schritt 2: IP-Adressen ermitteln (auf beiden VMs)

```
$ ip a
```

- IPv4-Adressen stehen hinter `inet`.
- IPv6-Adressen stehen hinter `inet6`.
- Wichtig ist die Schnittstelle, über die sich die VMs erreichen.

Schritt 3: Erreichbarkeit testen (auf VM1)

```
$ ping -c 3 <IP_von_VM2>
```

`-c 3` (count = Anzahl) sendet genau drei Pakete. Kommen Antworten? Dann ist VM2 im Netz erreichbar.

Schritt 4: Verbinden (auf VM1)

```
$ ssh benutzername@<IP_von_VM2>
```

Beim ersten Mal zeigt SSH einen Fingerabdruck des Servers. Er heißt **Fingerprint**. Er ist ein Kennzeichen für den Schlüssel des Servers. Du prüfst ihn und bestätigst mit `yes`. Danach gibst du das Passwort ein.

> ⚠️ **Sicherheitsregel:** Bestätige nicht jeden Fingerprint blind. Ein falscher Fingerprint kann auf einen Angreifer hindeuten. Der Fachbegriff dafür ist Man-in-the-Middle.

➕ Ergänzung: Der bestätigte Fingerprint wird in `~/.ssh/known_hosts` gespeichert. Ändert er sich später, warnt SSH deutlich.

Schritt 5: Auf VM2 arbeiten

```
$ cd ~/vm2_test
$ ls
hallo_vm2.txt
$ cat hallo_vm2.txt
Ich bin eine Textdatei auf VM2
```

Du sitzt an VM1. Du arbeitest aber im Dateisystem von VM2. Mit `exit` beendest du die Sitzung.

📝 Übungsfragen Block 6

Ü6.1 Warum soll man für Fernwartung SSH statt Telnet verwenden? Nennen Sie den Standard-Port von SSH.

Ü6.2 `ssh admin@10.0.0.16` bricht mit “Connection refused” ab. `ping 10.0.0.16` funktioniert. Nennen Sie zwei mögliche Ursachen und je einen Prüfschritt.

Ü6.3 Erklären Sie den Unterschied zwischen `systemctl start ssh` und `systemctl enable ssh`.

✅ Lösungen

Block 1

Ü1.1 Die Shell ist ein Programm. Sie nimmt Befehle entgegen, interpretiert sie und führt sie aus. Sie ist die Schnittstelle zwischen Benutzer und Betriebssystem. Reihenfolge: Benutzer → Shell → Kernel → Hardware.

Ü1.2 Pflicht sind `<quelle>` und `<ziel>` (spitze Klammern). `[optionen]` darf fehlen (eckige Klammern).

Ü1.3 Ja, es sind zwei Dateien. Linux unterscheidet bei Dateinamen Groß- und Kleinschreibung (case-sensitive).

Block 2

Ü2.1 Danach steht nur noch “Neustart” in der Datei. `>` überschreibt den alten Inhalt. Anhängen geht mit `echo Neustart >> log.txt`.

Ü2.2 Das Verzeichnis ist nicht leer. `rmdir` löscht nur leere Verzeichnisse. Mit `rm -r projekt` wird es samt Inhalt gelöscht. ⚠️ Achtung: Es gibt keinen Papierkorb. Prüfe vorher den Inhalt, z. B. mit `ls -la projekt`.

Ü2.3 Zwei von diesen:

- schnell leere Testdateien anlegen,
- Zeitstempel aktualisieren, damit ein zeitgesteuertes Backup die Datei erfasst,
- Neuübersetzung mit `make` erzwingen,
- Skripte testen, die nach Änderungsdatum suchen.

Block 3

Ü3.1 `ls -lah` (kombinierte Kurzoptionen) und `ls -l --all --human-readable` (mit Langoptionen). Auch die Reihenfolge darf wechseln, z. B. `ls -hal`.

Ü3.2 `apropos partition`. Es sucht in den Kurzbeschreibungen aller Handbuchseiten.

Ü3.3 `cd` ist in die Bash eingebaut (Builtin). Es hat oft keine eigene Manpage. Hilfe gibt es mit `help cd`.

Block 4

Ü4.1 Mit Strg + R die Rückwärtssuche öffnen. Dann `find` eingeben. Weitere Treffer mit erneutem Strg + R. Alternativ `history` nutzen.

Ü4.2

1. `nano ~/.bashrc` öffnen.
2. Zeile `alias la='ls -la'` eintragen.
3. Speichern mit Strg + O, Enter; beenden mit Strg + X.
4. `source ~/.bashrc` ausführen oder ein neues Terminal öffnen.

Block 5

Ü5.1 Esc drücken. So bist du sicher im Befehlsmodus. Dann `:q!` eingeben und mit Enter bestätigen.

Ü5.2 vi ist auf fast jedem Unix- und Linux-System vorhanden. Auf kleinen Servern oder Rettungssystemen fehlt nano manchmal. Dann ist vi der einzige Editor.

Block 6

Ü6.1 Telnet überträgt alles im Klartext, auch Passwörter. Ein Angreifer im Netz kann mitlesen. SSH verschlüsselt die gesamte Verbindung. Standard-Port: **TCP 22**.

Ü6.2 Das Netz funktioniert. Aber der Port antwortet nicht. Mögliche Ursachen (zwei davon):

- Der SSH-Dienst läuft nicht. Prüfen auf dem Server mit `systemctl status ssh`.
- Der SSH-Server ist nicht installiert. Prüfen bzw. installieren mit `apt install openssh-server`.
- Der Server lauscht auf einem anderen Port. Mit `ssh -p <port>` testen und die Serverkonfiguration prüfen.

ℹ️ Hinweis: Eine Firewall, die Pakete verwirft, führt eher zu einem Timeout. “Connection refused” heißt meist: Auf dem Port lauscht kein Dienst.

Ü6.3 `start` startet den Dienst sofort. Das gilt aber nur bis zum nächsten Neustart. `enable` sorgt dafür, dass der Dienst bei jedem Systemstart automatisch startet. Sofort startet er ihn aber nicht.

🎓 Schlagwortliste für die Prüfung

Begriff	Kurzdefinition	Block
Shell	Programm, das Befehle interpretiert; Hülle um den Kernel	1
Kernel	Kern des Betriebssystems, steuert die Hardware	1

Begriff	Kurzdefinition	Block
Bash	Bourne Again Shell, Standard-Shell unter Debian	1
Prompt `$` / `#`	Eingabebereitschaft als Benutzer / als Root	1
`[...]` / `<...>`	optionaler Teil / Pflichtangabe in der Syntax	1
case-sensitive	Groß- und Kleinschreibung wird unterschieden	1
echo	Text ausgeben	2
Umleitung `>` / `>>`	in Datei schreiben (überschreiben) / anhängen	2
ls, `-l`, `-a`	Inhalt auflisten, lange Form, auch versteckte Dateien	2
cat	Dateiinhalt anzeigen, Dateien aneinanderhängen	2
touch	leere Datei anlegen oder Zeitstempel aktualisieren	2
cp / cp -r	kopieren / Verzeichnis rekursiv kopieren	2
mv	verschieben oder umbenennen	2
rm / rm -r	löschen ohne Papierkorb / Verzeichnis samt Inhalt	2
mkdir / rmdir	Verzeichnis anlegen / leeres Verzeichnis löschen	2
cd, `cd ..`, `cd`	wechseln, eine Ebene hoch, ins Home-Verzeichnis	2
Kurzoption / Langoption	`-v` / `--verbose`	3
man	ausführliches Handbuch	3
`--help`	Kurzübersicht eines Befehls	3
help	Hilfe zu Bash-Builtins wie `cd`	3
apropos	Handbuchseiten nach Stichwort suchen	3
history / Strg + R	Befehlshistorie / Rückwärtssuche	4
Tab-Vervollständigung	Befehle und Pfade automatisch ergänzen	4
Terminal vs. Shell	Fenster-Programm vs. Befehlsinterpreter	4
less	lange Dateien seitenweise anzeigen	4
alias	eigener Kurzbefehl, dauerhaft in `~/.bashrc`	4
source	Datei in der laufenden Shell neu einlesen	4

Begriff	Kurzdefinition	Block
vi / vim	klassischer Editor / vi improved; `:wq`, `:q!`	5
nano	einfacher Editor; Strg + O speichern, Strg + X beenden	5
Befehlsmodus / Einfügemodus	die zwei Grundmodi von vi	5
SSH	verschlüsselte Fernverbindung, Standard TCP 22	6
Telnet	veraltete Fernverbindung im Klartext	6
SSH-Client / SSH-Server	baut Verbindung auf / nimmt Verbindungen an	6
`-p`, `-i`, `-v`	Port, Schlüsseldatei, ausführliche Meldungen	6
Fingerprint	Kennzeichen des Server-Schlüssels, vor Bestätigung prüfen	6
known_hosts	Datei mit bestätigten Server-Fingerprints	6
openssh-server	Paket für den SSH-Server unter Debian	6
systemctl start / enable / status	Dienst jetzt starten / beim Boot starten / Zustand prüfen	6
ip a	IP-Adressen der Schnittstellen anzeigen	6
ping -c	Erreichbarkeit testen, Anzahl Pakete festlegen	6

🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug

📘 Grundlage: Folien “Linux Modul 3: Grundlagen III” (GBS Fachschule). Die Zusammenfassung folgt den fünf Blöcken der Folien.

- **Fett** gedruckte Begriffe sind Prüfungs-Schlagwörter. Diese Begriffe solltest du erklären können.
- 🎯 Jeder Abschnitt hat eine Zeile “IHK-Relevanz”. Sie ist eine eigene Einschätzung. Sie ist keine amtliche Angabe.
- Der IHK-Leitfaden nennt Linux-Befehle nicht mit Namen. Linux passt aber zu diesen Prüfungsteilen:
  - **AP1** (Teil 1, “Einrichten eines IT-gestützten Arbeitsplatzes”): Betriebssysteme, IT-Sicherheit am Arbeitsplatz.
  - **AP2 Konzeption und Administration**: Systeme verwalten, Rechte verwalten, Aufgaben automatisieren (Skripte).
  - **AP2 Netzwerke**: nur am Rand. Beispiele: IP-Forwarding oder Netzwerk-Einstellungen in `/etc`.
- ✏️ Am Ende jedes Blocks stehen Übungsfragen. Die Lösungen stehen gesammelt am Ende.

💡 Das Zeichen vorne in den Beispielen heißt Prompt. `$` = normaler Benutzer. `#` = Root.

Symbol	Bedeutung
🎯	IHK-Relevanz
💡	Merksatz
⚠️	Achtung, Falle
🔒	Rechte und Sicherheit
📁	Ordner und Pfade
🧮	Rechenaufgabe
✏️	Übungsfragen
✅	Lösungen

👑 Block 1: Der Root-User und sudo

1.1 🔑 Root und sudo – die Idee IHK mittel

> 🎯 **IHK: mittel** Rechtevergabe und Sicherheit sind Kern von AP2 Konzeption/Administration; in AP1 eher als allgemeines Sicherheitsprinzip

Stell dir ein Gebäude vor. Der Hausmeister hat einen Generalschlüssel. Er kommt in jeden Raum. Unter Linux heißt dieser “Hausmeister” **Root**. Root ist der **Superuser**.

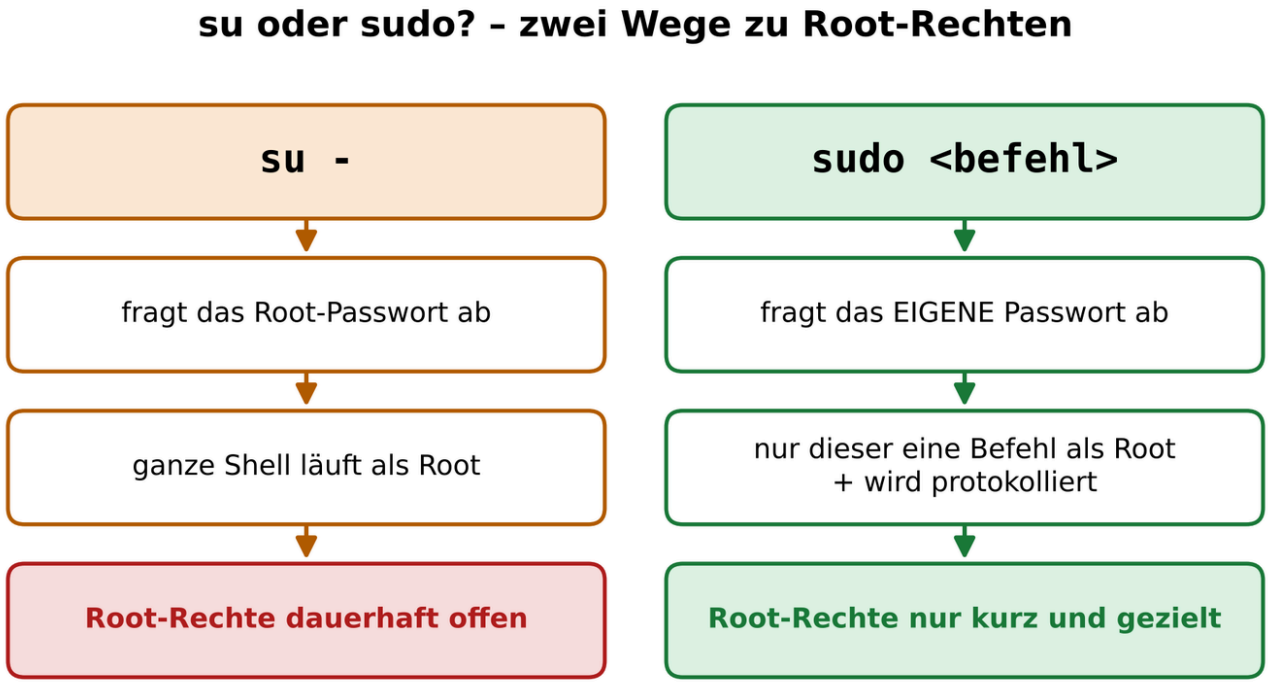
Root darf alles. Root darf jede Datei lesen, ändern und löschen. Unter Windows ist das wie das eingebaute Administrator-Konto.

⚠️ Arbeitet man immer als Root, ist das gefährlich. Ein Tippfehler kann das System beschädigen. Darum arbeitet man im Alltag als normaler Benutzer.

Root-Rechte holt man sich nur kurz für einen Befehl. Das Werkzeug dafür heißt **sudo**. Das ist wie: Generalschlüssel kurz ausleihen, Tür öffnen, Schlüssel zurückgeben.

💡 Merksatz: Root ist das Konto mit den höchsten Rechten. sudo ist das Werkzeug für einzelne Befehle mit diesen Rechten.

Weg zu Root-Rechten	Was passiert	Voraussetzung
Direkt als Root anmelden	Ganze Sitzung als Root	Oft gesperrt oder nicht empfohlen
`su` / `su -`	Wechsel in eine Root-Shell	Root-Passwort
`sudo <befehl>`	Ein einzelner Befehl läuft als Root	Benutzer ist für sudo berechtigt, eigenes Passwort
`sudo -i`	Root-Shell über das eigene Konto	Benutzer ist für sudo berechtigt



su und sudo im Vergleich: su öffnet dauerhaft eine Root-Shell, sudo führt nur einen Befehl als Root aus

su und sudo im Vergleich: su öffnet dauerhaft eine Root-Shell, sudo führt nur einen Befehl als Root aus

📖 Die Namen:

- `su` = **substitute user** oder switch user. Das heißt: zu einem anderen Benutzer wechseln. ⚠️ Es heißt nicht “super user”.
- `sudo` = substitute user do oder superuser do. Das heißt: einen Befehl als anderer Benutzer ausführen.

✅ Vorteile von sudo gegenüber su:

1. **Feine Rechtevergabe**: Man legt fest, wer welche Befehle als Root ausführen darf.
2. **Nachvollziehbarkeit**: sudo schreibt mit, wer welchen Befehl ausgeführt hat.
3. **Mehr Sicherheit im Alltag**: Root-Rechte gibt es nur kurz. Und nur, wenn man sie braucht.

1.2 🛠️ sudo-Rechte vergeben IHK mittel

> 🎯 **IHK: mittel** typisches Administrationsthema für AP2 Konzeption/Administration (“Benutzer- und Rechteverwaltung”)

Auf manchen Systemen fehlt sudo. So installiert man es unter Debian:

```
$ su -
# apt install sudo
```

Variante 1: Benutzer in die Gruppe sudo aufnehmen (Debian/Ubuntu)

Eine Gruppe ist wie ein Team. Wer im Team “sudo” ist, darf sudo nutzen.

Befehl	Wirkung
`# usermod -aG sudo max`	max kommt in die Gruppe sudo (`-a` = anhängen, `-G` = Gruppe)
`# usermod -rG sudo max`	max wird aus der Gruppe sudo entfernt

⚠️ Wichtig: Danach muss sich der Benutzer neu anmelden. Erst dann gilt die neue Gruppe.

Variante 2: Nur einzelne Befehle erlauben

Das ist wie ein Schlüssel für genau eine Tür. Geregelt wird das in der **sudoers**-Konfiguration. Zusätzliche Regeln liegen in `/etc/sudoers.d/`. Das `.d` steht für directory, also Verzeichnis.

Man bearbeitet diese Dateien immer mit **visudo**:

```
# visudo -f /etc/sudoers.d/max
```

visudo macht zwei Dinge:

1. 🔒 Es sperrt die Datei. So kann niemand gleichzeitig daran arbeiten.
2. ✔️ Es prüft die Schreibweise (Syntax), bevor die Änderung gilt. Sonst kann ein Tippfehler sudo komplett blockieren.

Das “vi” im Namen ist historisch. Auf Debian öffnet sich oft nano.

🔍 **Beispiel: eine sudoers-Zeile zerlegen**

```
max ALL=(root) NOPASSWD: /usr/bin/systemctl restart apache2
```

Teil	Bedeutung
`max`	Benutzer, für den die Regel gilt
`ALL`	Regel gilt auf allen Hosts
`(root)`	Befehl läuft als root
`NOPASSWD:`	keine Passwortabfrage
`/usr/bin/systemctl restart apache2`	genau dieser Befehl ist erlaubt

Es gibt auch eine sehr weite Variante: `max ALL=(ALL:ALL) NOPASSWD: ALL`. Sie erlaubt alle Befehle. Sie gilt als jeder Benutzer und jede Gruppe. Ein Passwort ist nicht nötig. Das nutzt man nur in Laboren oder für kontrollierte Automatisierung.

⚠️ Fallstrick: sudoers-Regeln sind **additiv**. Additiv heißt: Sie zählen zusammen. Ist max zusätzlich in der Gruppe sudo, hat er trotzdem alle Rechte. Eine Einzelregel schränkt also nichts ein.

1.3 💲 Der Shell-Prompt IHK gering

> 🎯 **IHK: gering** Grundwissen; hilft beim Lesen von Aufgaben, wird aber kaum einzeln abgefragt

Der Prompt ist die Eingabezeile. Das letzte Zeichen verrät, wer gerade arbeitet.

Zeichen am Ende	Bedeutung
`$`	normaler Benutzer, z. B. `user@hostname:~$`
`#`	Root, z. B. `root@hostname:~#`
`%`	normaler Benutzer bei der Shell zsh
`>`	Fortsetzungsprompt: Die Shell wartet auf den Rest der Eingabe

Beispiel für den Fortsetzungsprompt: Ein Anführungszeichen ist noch offen.

```
$ echo "Hallo
>
```

✏️ Übungsfragen Block 1

Ü1.1 Nennen Sie zwei Vorteile von `sudo` gegenüber `su`.

Ü1.2 Erklären Sie die Zeile `anna ALL=(root) NOPASSWD: /usr/bin/systemctl restart nginx`. Darf anna damit auch `apt update` als Root ausführen?

Ü1.3 Ein Kollege bearbeitet `/etc/sudoers` direkt mit nano. Welches Risiko besteht, und welches Werkzeug sollte er nutzen?

🔒 Block 2: Dateirechte

2.1 📋 Grundlagen und `ls -l` lesen IHK hoch

> 🎯 **IHK: hoch** Dateirechte sind Kernwissen für AP2 Konzeption/Administration; in AP1 möglich als Teil von “Betriebssysteme” und IT-Sicherheit

Linux ist ein **Multiuser-Betriebssystem**. Das heißt: Viele Benutzer arbeiten am selben System. Dateirechte legen fest, wer was mit einer Datei tun darf. Das ist wie bei einem Spind: Wer darf ihn öffnen, wer darf etwas hineinlegen?

Es gibt drei Grundrechte:

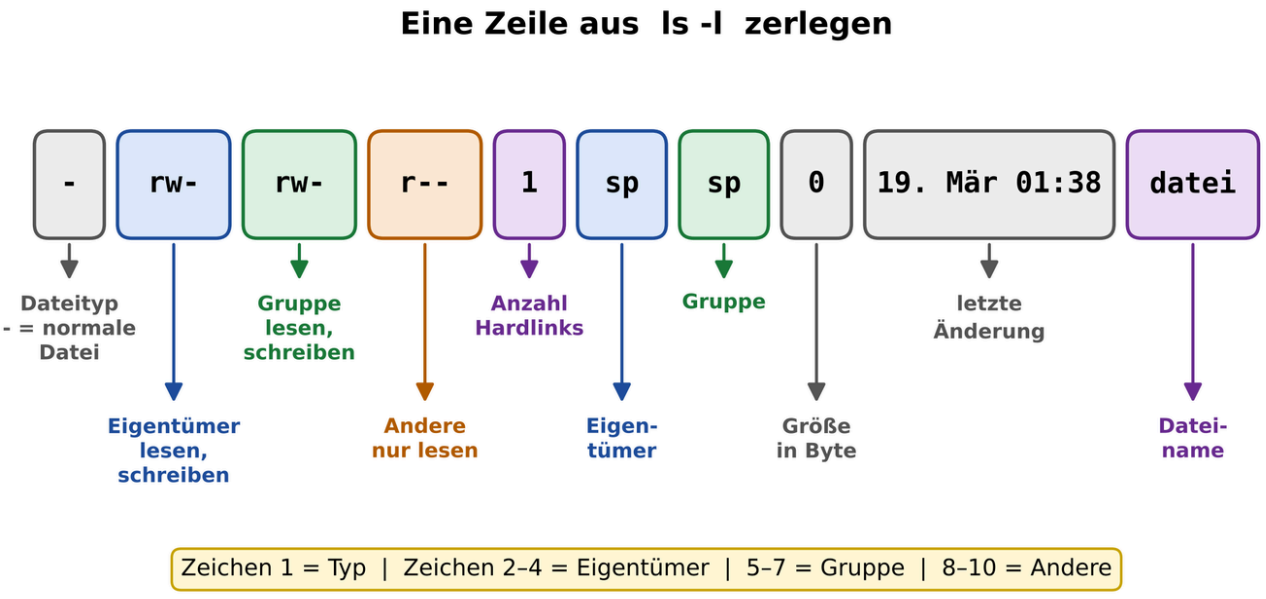
Recht	Buchstabe	Bei Dateien	Bei Verzeichnissen
👁️ Lesen	`r` (read)	Inhalt lesen	Inhalt auflisten
✍️ Schreiben	`w` (write)	Inhalt ändern	Dateien anlegen, löschen, umbenennen
▶️ Ausführen	`x` (execute)	als Programm starten	Verzeichnis betreten bzw. durchsuchen

Die Rechte gelten für drei Klassen:

- 👤 **Eigentümer** (user, `u`)
- 👥 **Gruppe** (group, `g`)
- 🌍 **Andere** (others, `o`)

🔍 Beispiel: eine Zeile aus `ls -l` lesen

```
-rw-rw-r-- 1 sp sp 0 19. Mär 01:38 datei
```



Die Zeile aus ls -l in ihre Teile zerlegt: Dateityp, Rechte für Eigentümer, Gruppe und Andere, Hardlinks, Eigentümer, Gruppe, Größe, Datum, Name

Die Zeile aus ls -l in ihre Teile zerlegt: Dateityp, Rechte für Eigentümer, Gruppe und Andere, Hardlinks, Eigentümer, Gruppe, Größe, Datum, Name

Teil	Wert	Bedeutung
Zeichen 1	`-`	Dateityp: normale Datei
Zeichen 2–4	`rw-`	Eigentümer: lesen, schreiben
Zeichen 5–7	`rw-`	Gruppe: lesen, schreiben
Zeichen 8–10	`r--`	Andere: nur lesen
`1`		Anzahl der **Hardlinks**
`sp`	erstes	Eigentümer
`sp`	zweites	Gruppe
`0`		Größe in Byte
`19. Mär 01:38`		letzte Änderung
`datei`		Dateiname

🗂️ **Dateitypen** (erstes Zeichen):

Zeichen	Typ
`-`	normale (reguläre) Datei
`d`	Verzeichnis (directory)
`l`	symbolischer Link
`c`	Zeichen-Gerätedatei (character device), z. B. `/dev/tty`
`b`	Block-Gerätedatei (block device), z. B. Festplatten
`s`	Socket
`p`	Named Pipe (FIFO): Datenaustausch zwischen zwei Prozessen

2.2 🔗 Hardlinks und symbolische Links IHK mittel

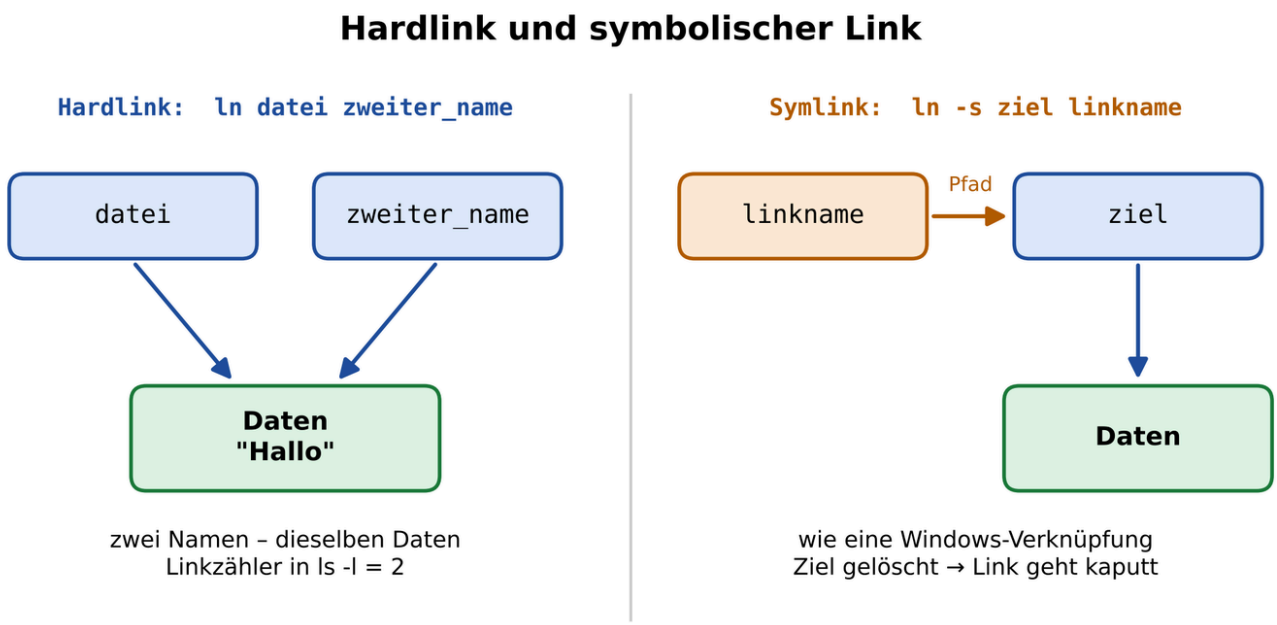
> 🎯 **IHK: mittel** Unterschied Hardlink/Symlink ist ein klassisches Abgrenzungsthema für AP2 Konzeption/Administration

Ein Mensch kann zwei Namen haben, z. B. Vorname und Spitzname. Es bleibt dieselbe Person. So ist es beim **Hardlink**: Er ist ein weiterer Name für denselben Dateiinhalt. Die Zahl in `ls -l` zeigt, wie viele Namen auf den Inhalt zeigen.

```
$ echo "Hallo" > datei
$ ln datei zweiter_name
$ ls -l datei zweiter_name
```

Beide Einträge zeigen jetzt die Zahl 2. Es sind keine Kopien. Es sind zwei Namen für dieselben Daten.

Ein **symbolischer Link** (Symlink) ist dagegen wie ein Wegweiser. Er zeigt nur auf den Pfad einer Datei.



Hardlink: zwei Namen zeigen auf dieselben Daten. Symlink: ein Link zeigt auf den Pfad einer anderen Datei

Hardlink: zwei Namen zeigen auf dieselben Daten. Symlink: ein Link zeigt auf den Pfad einer anderen Datei

Symbolischer Link (Symlink)	Hardlink
zeigt auf den Pfad einer Datei	zeigt direkt auf dieselben Daten
wie eine Windows-Verknüpfung	wie ein zweiter echter Dateiname
geht kaputt, wenn das Ziel gelöscht oder verschoben wird	bleibt gültig, wenn ein anderer Name gelöscht wird
kann auf Verzeichnisse zeigen	normalerweise nicht auf Verzeichnisse
geht über Dateisystemgrenzen hinweg	nur im selben Dateisystem

Ein Symlink entsteht mit `ln -s ziel linkname` (Ergänzung, nicht auf den Folien).

2.3 🧱 chmod – oktal und symbolisch IHK hoch

> 🎯 **IHK: hoch** Oktalwerte umrechnen ist eine typische Kurzaufgabe mit Rechenweg; passt zu AP2 Konzeption/Administration

Befehl	Zweck	Wer darf?
`chmod`	Rechte ändern (change mode)	Eigentümer oder Root
`chown`	Eigentümer ändern (change owner)	normalerweise nur Root
`chgrp`	Gruppe ändern (change group)	Eigentümer (nur auf eigene Gruppen) oder Root

🔢 Oktale Schreibweise

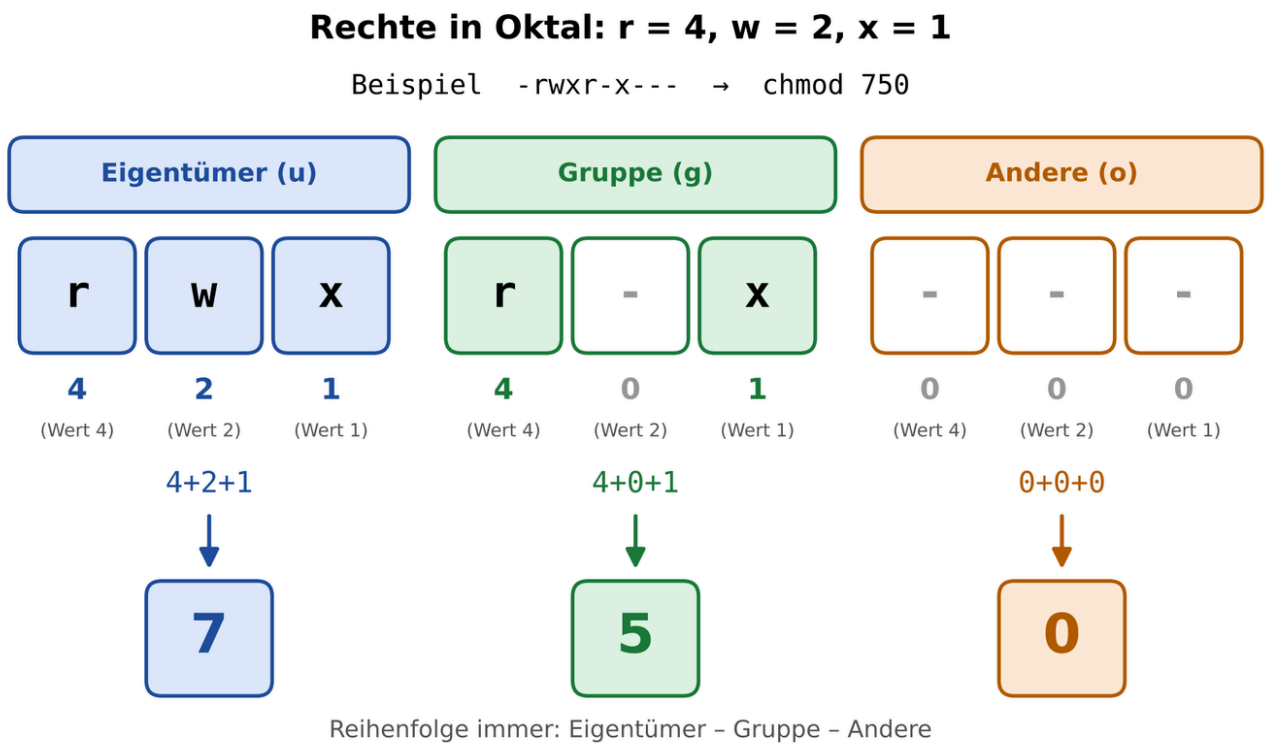
Jedes Recht hat einen Zahlenwert: **r = 4, w = 2, x = 1**. Pro Klasse zählt man die Werte zusammen. So entsteht eine Ziffer von 0 bis 7. Das ist wie Münzen zählen: Eine 4er-, eine 2er- und eine 1er-Münze.

Ziffer	Rechnung	Rechte
7	4+2+1	`rwx`
6	4+2	`rw-`
5	4+1	`r-x`
4	4	`r--`
3	2+1	`-wx`
2	2	`-w-`
1	1	`--x`
0	0	`---`

💡 Reihenfolge der drei Ziffern: Eigentümer – Gruppe – Andere.

🧱 Beispiel 1: Rechte in Oktal umrechnen

Gegeben: `-rwxr-x---`



Rechenschema: rwx ergibt 4+2+1 = 7, r-x ergibt 4+0+1 = 5, — ergibt 0. Ergebnis 750

Rechenschema: rwx ergibt 4+2+1 = 7, r-x ergibt 4+0+1 = 5, — ergibt 0. Ergebnis 750

Klasse	Rechte	Rechnung	Ziffer
Eigentümer	`rwx`	4+2+1	7
Gruppe	`r-x`	4+0+1	5
Andere	`---`	0+0+0	0

Ergebnis: **750**, also `chmod 750 datei`.

Umgekehrt: `chmod 644 datei.txt` ergibt 6 = `rw-`, 4 = `r--`, 4 = `r--`. Also `-rw-r--r--`.

🔤 Symbolische Schreibweise

Hier schreibt man Buchstaben statt Zahlen. Syntax: `chmod [ugoa][+-=][rwx] datei`

Teil	Bedeutung
`u`, `g`, `o`, `a`	Eigentümer, Gruppe, Andere, alle (a = u+g+o)
`+`	Recht hinzufügen, Rest bleibt
`-`	Recht entfernen, Rest bleibt
`=`	Rechte der Klasse neu setzen, alte Rechte dieser Klasse werden ersetzt

🧱 Beispiel 2: symbolisch ändern

Ausgangslage: `-rw-r-----` (oktal 640).

```
$ chmod g+w,o+r datei
```

- Gruppe: `r--` plus `w` ergibt `rw-` = 6.
- Andere: `---` plus `r` ergibt `r--` = 4.
- Eigentümer bleibt `rw-` = 6.

Ergebnis: `-rw-rw-r--` = **664**.

Danach `chmod u=rwx,g=rx,o= datei`: Alle drei Klassen werden neu gesetzt. Ergebnis `-rwxr-x---` = **750**.

💡 Wann was?

- **Oktal** setzt immer alle Rechte auf einmal. Gut für Vorgaben, Dokumentation und Skripte: `chmod 755 script.sh`.
- **Symbolisch** ändert gezielt einzelne Rechte. Gut, wenn der Rest bleiben soll: `chmod u+x script.sh`.

2.4 👤 chown und chgrp IHK mittel

> 🎯 **IHK: mittel** gehört zur Benutzer- und Rechteverwaltung in AP2 Konzeption/Administration

```
# chown sebastian dump.sql           # nur Eigentümer ändern
# chown sebastian:admins dump.sql    # Eigentümer und Gruppe in einem Schritt
$ chgrp projekt bericht.txt          # nur Gruppe ändern
```

Ein normaler Benutzer darf `chgrp` nur auf eigene Dateien anwenden. Er darf nur Gruppen setzen, in denen er selbst Mitglied ist.

2.5 🎭 umask – Standardrechte für neue Dateien IHK hoch

> 🎯 **IHK: hoch** die umask-Rechnung ist eine gut prüfbare Rechenaufgabe für AP2 Konzeption/Administration

Stell dir eine Schablone vor. Sie deckt Teile ab. Genau so arbeitet die **umask** (user mask). Sie legt fest, welche Rechte bei neuen Dateien und Verzeichnissen **entfernt** werden.

Ausgangsrechte, bevor die umask wirkt:

- 📄 Dateien: **666** (`rw-rw-rw-`). Neue Dateien sind nie automatisch ausführbar.
- 📁 Verzeichnisse: **777** (`rwxrwxrwx`). Ohne `x` könnte man ein Verzeichnis nicht betreten.

🧱 Beispiel 3: umask 022

	Datei	Verzeichnis
Ausgangsrechte	666 = `rw-rw-rw-`	777 = `rwxrwxrwx`
umask	022 = bei Gruppe und Anderen `w` entfernen	022
Ergebnis	644 = `rw-r--r--`	755 = `rwxr-xr-x`

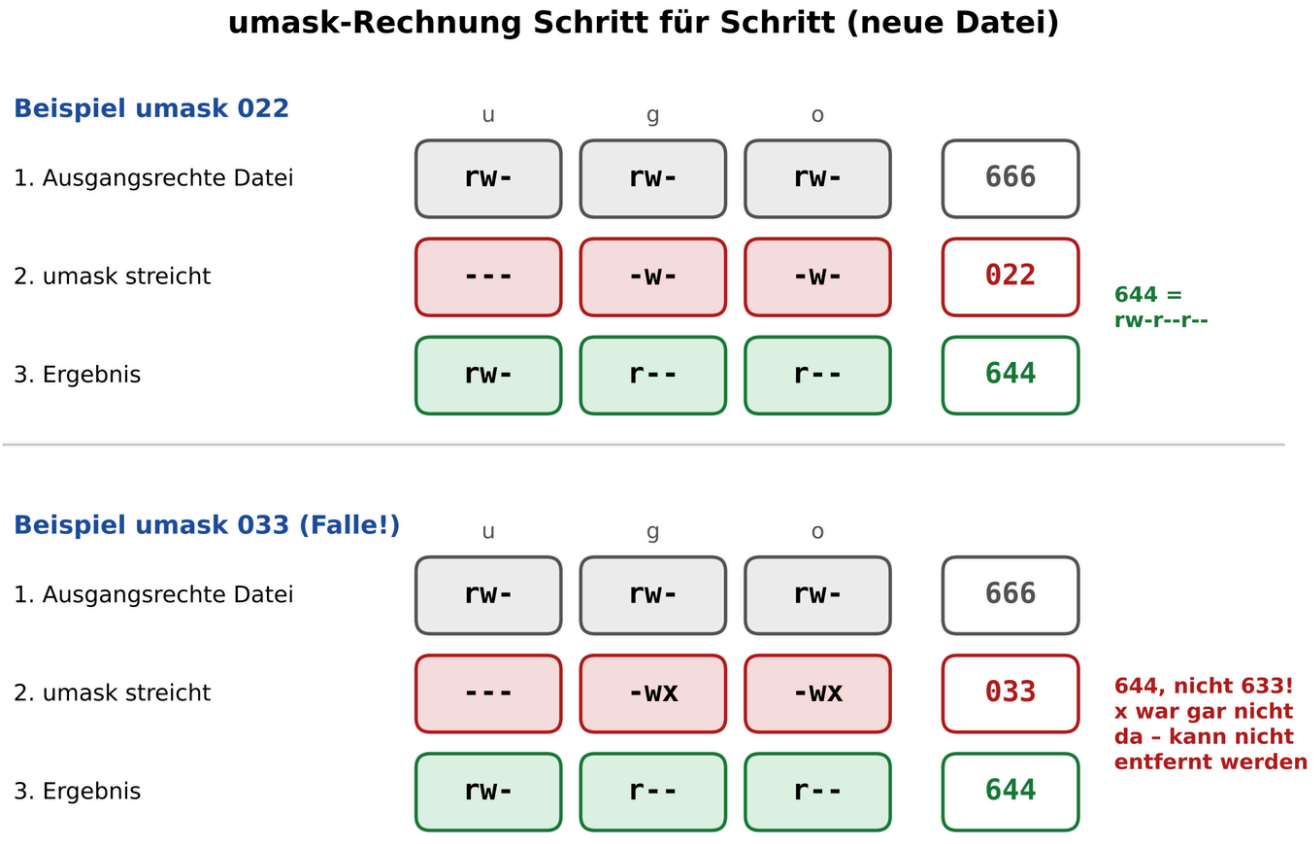
⚠️ **Wichtiger Hinweis zur Rechnung:** Die Folien rechnen "666 − 022 = 644". Das klappt bei den üblichen Werten. Genau genommen **entfernt** die umask aber Rechte-Bits. Sie zieht nicht Stelle für Stelle ab. Ein Recht, das gar nicht da ist, kann man nicht entfernen.

🧱 Beispiel 4: umask 033 (Falle bei reiner Subtraktion)

Für Dateien:

- Ausgangsrechte Gruppe: 6 = `rw-`. Die umask-Ziffer 3 = `-wx` sagt: `w` und `x` entfernen.
- `rw-` ohne `w` und `x` ergibt `r--` = 4. Dasselbe gilt für Andere.
- Ergebnis: **644** (`rw-r--r--`), nicht 633.

❌ Falsch wäre: 666 − 033 = 633. Dann hätte die Gruppe `-wx`. Das stimmt nicht.



umask Schritt für Schritt: bei 022 und bei 033 entsteht für eine neue Datei jeweils 644, weil x vorher nicht vorhanden war

umask Schritt für Schritt: bei 022 und bei 033 entsteht für eine neue Datei jeweils 644, weil x vorher nicht vorhanden war

Für Verzeichnisse: 777 mit umask 033 ergibt **744** (`rwxr--r--`). Hier stimmt die Subtraktion. Denn hier sind alle Bits vorhanden.

💡 Sicherer Rechenweg in drei Schritten:

1. Ausgangsrechte als Buchstaben schreiben.
2. Pro Klasse die Rechte streichen, die in der umask-Ziffer stehen.
3. Übrige Rechte wieder in Oktal umrechnen.

📊 **Häufige umask-Werte** (alle nachgerechnet)

umask	Bedeutung	Neue Dateien	Neue Verzeichnisse
000	nichts entfernen	666 `rw-rw-rw-`	777 `rwxrwxrwx`
002	`w` bei Anderen entfernen	664 `rw-rw-r--`	775 `rwxrwxr-x`
022	`w` bei Gruppe und Anderen entfernen	644 `rw-r--r--`	755 `rwxr-xr-x`
027	`w` bei Gruppe, alles bei Anderen entfernen	640 `rw-r-----`	750 `rwxr-x---`
077	alles bei Gruppe und Anderen entfernen	600 `rw-------`	700 `rwx------`

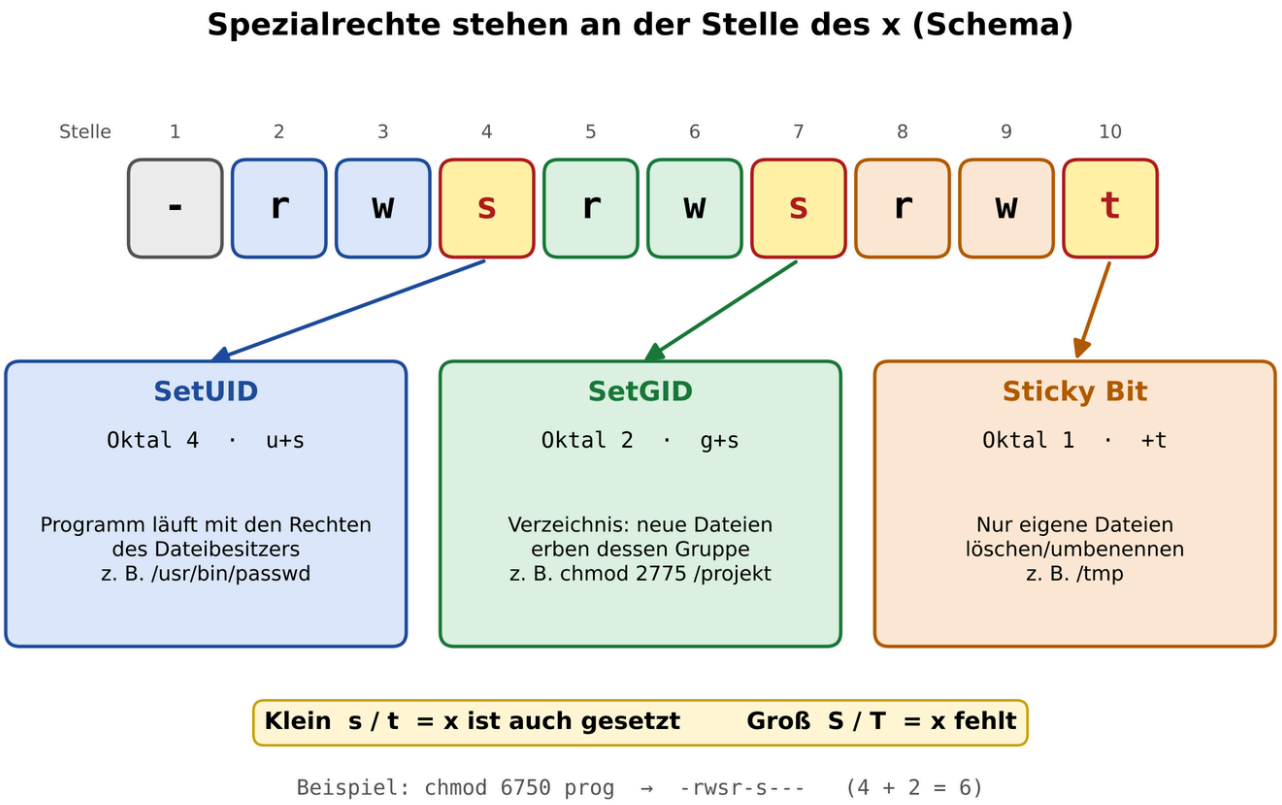
Befehl	Wirkung
`umask`	aktuelle Maske anzeigen
`umask 027`	neue Maske setzen, gilt nur für die aktuelle Shell-Sitzung
`umask -S`	Maske symbolisch anzeigen; bei 022: `u=rwx,g=rx,o=rx`

Dauerhaft setzt man die umask in Konfigurationsdateien. Für das ganze System unter Debian: `/etc/login.defs`. Für einen Benutzer z. B. in `~/.profile`.

2.6 ⭐ Spezialrechte: SetUID, SetGID, Sticky Bit IHK mittel bis hoch

> 🎯 **IHK: mittel bis hoch** das Erkennen von s und t in ls -l ist eine typische Kurzfrage in AP2 Konzeption/Administration; Sicherheitsbezug

Neben r, w und x gibt es drei Sonderrechte. Sie stehen dort, wo sonst ein `x` steht.



Schema: SetUID steht an Stelle 4, SetGID an Stelle 7, Sticky Bit an Stelle 10 der Rechteanzeige

Schema: SetUID steht an Stelle 4, SetGID an Stelle 7, Sticky Bit an Stelle 10 der Rechteanzeige

Spezialrecht	Oktal-Präfix	Symbolisch	Anzeige an Stelle	Wirkung
SetUID	4	`u+s`	4 (x des Eigentümers)	Programm läuft mit den Rechten des Dateibesitzers
SetGID	2	`g+s`	7 (x der Gruppe)	Verzeichnis: neue Dateien erben die Gruppe des Verzeichnisses. Programm: läuft mit der Gruppe der Datei
Sticky Bit	1	`+t`	10 (x der Anderen)	In gemeinsamen Verzeichnissen darf man nur eigene Dateien löschen oder umbenennen

Die Präfixe kann man addieren: 4 + 2 = 6 setzt SetUID und SetGID zugleich.

🔍 Typische Beispiele:

Beispiel	Anzeige	Erklärung
`/usr/bin/passwd`	`-rwsr-xr-x`	SetUID: Benutzer ändern ihr Passwort, das Programm schreibt als root in `/etc/shadow`
`chmod 2775 /projekt`	`drwxrwsr-x`	SetGID: alle neuen Dateien gehören zur Projektgruppe
`/tmp`	`drwxrwxrwt`	Sticky Bit: jeder darf anlegen, aber nur eigene Dateien löschen

Das Sticky Bit ist wie ein Gemeinschaftskühlschrank: Jeder darf etwas hineinstellen. Wegnehmen darf man nur sein eigenes Essen. In einem Verzeichnis mit Sticky Bit dürfen nur drei Personen eine Datei löschen oder umbenennen: der Dateibesitzer, der Besitzer des Verzeichnisses und Root.

🔠 Kleinbuchstabe oder Großbuchstabe?

Das Spezialrecht steht an der Stelle des `x`. Deshalb gibt es zwei Fälle:

Anzeige	Spezialrecht	x-Recht an dieser Stelle
`s`	SetUID bzw. SetGID gesetzt	x ist gesetzt
`S`	SetUID bzw. SetGID gesetzt	x ist **nicht** gesetzt
`t`	Sticky Bit gesetzt	x für Andere ist gesetzt
`T`	Sticky Bit gesetzt	x für Andere ist **nicht** gesetzt

⚠️ Ein großes `S` bei einer Datei deutet meist auf einen Fehler hin. SetUID ohne Ausführungsrecht hat keine Wirkung.

🧱 Beispiel 5: Spezialrechte erkennen und in Oktal umrechnen (mit Testsystem geprüft)

Befehl	Anzeige `ls -l`	Lesart
`chmod 4755 prog`	`-rwsr-xr-x`	SetUID, Eigentümer hat x
`chmod 4644 datei`	`-rwSr--r--`	SetUID, aber Eigentümer ohne x
`chmod 2755 verz`	`drwxr-sr-x`	SetGID, Gruppe hat x
`chmod 2640 datei`	`-rw-r-S---`	SetGID, aber Gruppe ohne x
`chmod 1777 verz`	`drwxrwxrwt`	Sticky, Andere haben x
`chmod 1770 verz`	`drwxrwx--T`	Sticky, Andere ohne x
`chmod 6750 prog`	`-rwsr-s---`	SetUID und SetGID (4+2 = 6)

Rückweg am Beispiel `drwxrws---`:

1. Grundrechte lesen. Das `s` zählt dabei als `x`: `rwx` `rwx` `---` = 770.
2. Spezialrechte: `s` bei der Gruppe = SetGID = 2.
3. Ergebnis: **2770**.

2.7 🛡️ Principle of Least Privilege und Zero Trust IHK hoch

> 🎯 **IHK: hoch** IT-Sicherheitsprinzipien sind in AP1 (IT-Sicherheit am Arbeitsplatz) und in beiden AP2-Bereichen prüfbar

Ein Praktikant bekommt nicht den Generalschlüssel. Er bekommt nur den Schlüssel für sein Büro. Das ist das **Principle of Least Privilege (PoLP)**. Auf Deutsch: Prinzip der minimalen Rechte. Benutzer und Prozesse bekommen nur die Rechte, die sie wirklich brauchen.

💡 Merksatz: So wenig Rechte wie möglich, so viele wie nötig.

Umsetzung unter Linux:

- Admin-Rechte nur bei Bedarf über sudo. Keine dauerhafte Root-Sitzung.
- Keine unnötigen Gruppen, z. B. `sudo`, `docker`, `adm`.
- Strenge Standardrechte durch eine passende umask.
- `x` nur dort setzen, wo es gebraucht wird.

Zero Trust heißt: Nichts wird automatisch vertraut. Jeder Zugriff wird immer wieder geprüft. Das ist wie eine Ausweiskontrolle an jeder Tür, nicht nur am Haupteingang. PoLP ist ein Baustein von Zero Trust.

- PoLP = minimale Rechte.
- Zero Trust = minimales Vertrauen plus ständige Überprüfung.

✏️ Übungsfragen Block 2

Ü2.1 `ls -l` zeigt: `-rwxr-x--- 2 max verwaltung 812 3. Feb 09:14 backup.sh`. Geben Sie den Oktalwert an. Was bedeutet die Zahl 2? Darf der Benutzer anna (nicht in der Gruppe verwaltung) das Skript ausführen?

Ü2.2 Ein Administrator setzt `umask 027`. Berechnen Sie die Rechte einer neuen Datei und eines neuen Verzeichnisses, oktal und symbolisch. Geben Sie den Rechenweg an.

Ü2.3 Ein Projektverzeichnis zeigt `drwxrws--T`. Welche Spezialrechte sind gesetzt? Wie lautet der vollständige Oktalwert?

📁 Block 3: Verzeichnisse

3.1 🌳 Verzeichnisbaum und FHS IHK mittel

> 🎯 **IHK: mittel** Aufbau des Dateisystems ist Basiswissen “Betriebssysteme” (AP1) und Voraussetzung für AP2 Konzeption/Administration

Linux ordnet alle Dateien in einem **Verzeichnisbaum**. Stell dir einen Baum auf dem Kopf vor. Oben ist die Wurzel, unten die Äste. Es gibt genau eine **Wurzel**: das Root-Verzeichnis `/`. Man spricht es “root” oder “slash”.

Linux	Windows
eine einzige Wurzel `/`	jedes Laufwerk ist eine eigene Wurzel (`C:\`, `D:\`)
weitere Datenträger werden in den Baum eingehängt	weitere Datenträger bekommen eigene Laufwerksbuchstaben

Jedes Verzeichnis hat genau ein übergeordnetes Verzeichnis. Es kann beliebig viele Unterverzeichnisse haben.

Der **FHS** (Filesystem Hierarchy Standard) ist ein Bauplan. Er legt fest, welche Standardverzeichnisse es gibt und wofür. Vorteil: Alle Distributionen sind ähnlich aufgebaut. Das macht die Verwaltung leichter.

Hintergrund: Viele Dateisysteme wie ext4 nutzen intern Varianten des **B-Baums**. Diese Baumstruktur hält Suchwege kurz. Denn alle Zweige sind gleich lang.

3.2 🗄️ Standardverzeichnisse IHK mittel

> 🎯 **IHK: mittel** Zuordnungsaufgaben (“Wo liegen Logdateien?”) sind gut prüfbar, v. a. AP2 Konzeption/Administration

Verzeichnis	Inhalt
`/`	Wurzel des gesamten Baums
`/bin`	Benutzerbefehle (bin = binary, ausführbare Programme)
`/sbin`	System- und Administratorbefehle
`/boot`	Kernel und Bootloader-Dateien
`/dev`	Gerätedateien (virtuell)

Verzeichnis	Inhalt
`/etc`	systemweite Konfigurationsdateien
`/home`	Home-Verzeichnisse der Benutzer
`/root`	Home-Verzeichnis von Root
`/lib`	Bibliotheken und Kernel-Module
`/media`	Einhängepunkt für Wechseldatenträger (USB, DVD)
`/mnt`	temporärer Einhängepunkt für manuell eingehängte Datenträger
`/opt`	zusätzlich installierte Softwarepakete
`/tmp`	temporäre Dateien, meist beim Neustart gelöscht
`/run`	Laufzeitdaten seit dem letzten Start (PID-Dateien, Sockets)
`/srv`	Daten von Diensten, z. B. Web- oder FTP-Daten
`/usr`	systemweite Programme, Bibliotheken, Ressourcen (“Userland”)
`/var`	veränderliche Daten, z. B. `/var/log` (Logdateien), `/var/cache`
`/proc`	virtuell: Prozess- und Systeminformationen
`/sys`	virtuell: Kernel- und Hardware-Schnittstellen

💡 Wichtige Merkpunkte:

- `/etc` kommt von “et cetera”. Heute liegen dort die Einstellungen für das ganze System.
- ⚠️ `/usr` heißt nicht “Benutzerdaten”. Die liegen in `/home`.
- **usrmerge**: Auf vielen neuen Systemen sind `/bin` und `/sbin` nur noch Symlinks. Sie zeigen auf `/usr/bin` und `/usr/sbin`. Prüfen mit `ls -ld /bin /sbin`.

3.3 👻 Virtuelle Verzeichnisse IHK gering bis mittel

> 🎯 **IHK: gering bis mittel** eher Hintergrundwissen; /proc und IP-Forwarding können in AP2 Netzwerke oder Konzeption auftauchen

`/dev`, `/proc` und `/sys` erzeugt der Kernel beim Systemstart. Sie liegen nicht auf der Festplatte. Der Kernel stellt die Inhalte im Arbeitsspeicher bereit, und zwar laufend neu. Das ist wie eine Anzeigetafel am Bahnhof: Sie zeigt immer den aktuellen Stand.

💡 Grundsatz unter Linux: Fast alles wird wie eine Datei behandelt. Deshalb funktionieren normale Befehle wie `cat`, `less` und `ls`.

Befehl	Ergebnis
`cat /proc/cpuinfo`	Informationen zur CPU
`cat /proc/meminfo`	Informationen zum Arbeitsspeicher
`cat /proc/version`	Kernel-Version
`cat /proc/sys/net/ipv4/ip_forward`	1 = IPv4-Forwarding an, 0 = aus
`echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward`	IPv4-Forwarding bis zum Neustart einschalten
`ls -l /dev`	Gerätedateien anzeigen

⚠️ **Warum `tee` und nicht `sudo echo 1 > datei`?**

- Bei `sudo echo 1 > datei` läuft nur `echo` als Root.
- Die Umleitung `>` macht die eigene Shell. Die hat keine Root-Rechte.
- Ergebnis: “Keine Berechtigung”.
- Bei `echo 1 | sudo tee datei` schreibt `tee` selbst in die Datei. Und `tee` läuft mit sudo.

`tee` gibt die Eingabe aus. Gleichzeitig schreibt es sie in eine Datei. Der Name kommt vom T-Stück aus der Rohrleitungstechnik.

Abgrenzung: `/run` enthält echte, aber temporäre Laufzeitdateien. `/proc` und `/sys` sind dagegen ein Live-Statusfenster des Systems.

3.4 👉 Verzeichniszeiger und versteckte Dateien IHK mittel

> 🎯 **IHK: mittel** und .. sind Voraussetzung für Pfadaufgaben; versteckte Dateien sind Basiswissen

Eintrag	Bedeutung
`.`	aktuelles Verzeichnis
`..`	übergeordnetes Verzeichnis
`.name`	**versteckte Datei** bzw. verstecktes Verzeichnis, z. B. `.bashrc`, `.ssh`, `.config`

`ls -la` zeigt alle Dateien, auch die versteckten.

✏️ Übungsfragen Block 3

Ü3.1 Ordnen Sie zu: a) Logdateien, b) systemweite Konfiguration, c) Home-Verzeichnis von Root, d) Laufzeitinformationen zu Prozessen.

Ü3.2 Ein Azubi behauptet: “ `/dev` enthält echte Dateien auf der Festplatte.” Nehmen Sie Stellung.

Ü3.3 Was bedeutet usrmerge, und mit welchem Befehl prüfen Sie es?

🧭 Block 4: Relative und absolute Pfade

4.1 📍 Absolute und relative Pfade IHK mittel

> 🎯 **IHK: mittel** Pfadangaben tauchen in jeder Shell- oder Skriptaufgabe auf (AP2 Konzeption/Administration), selten als eigene Frage

Ein Pfad ist wie eine Wegbeschreibung.

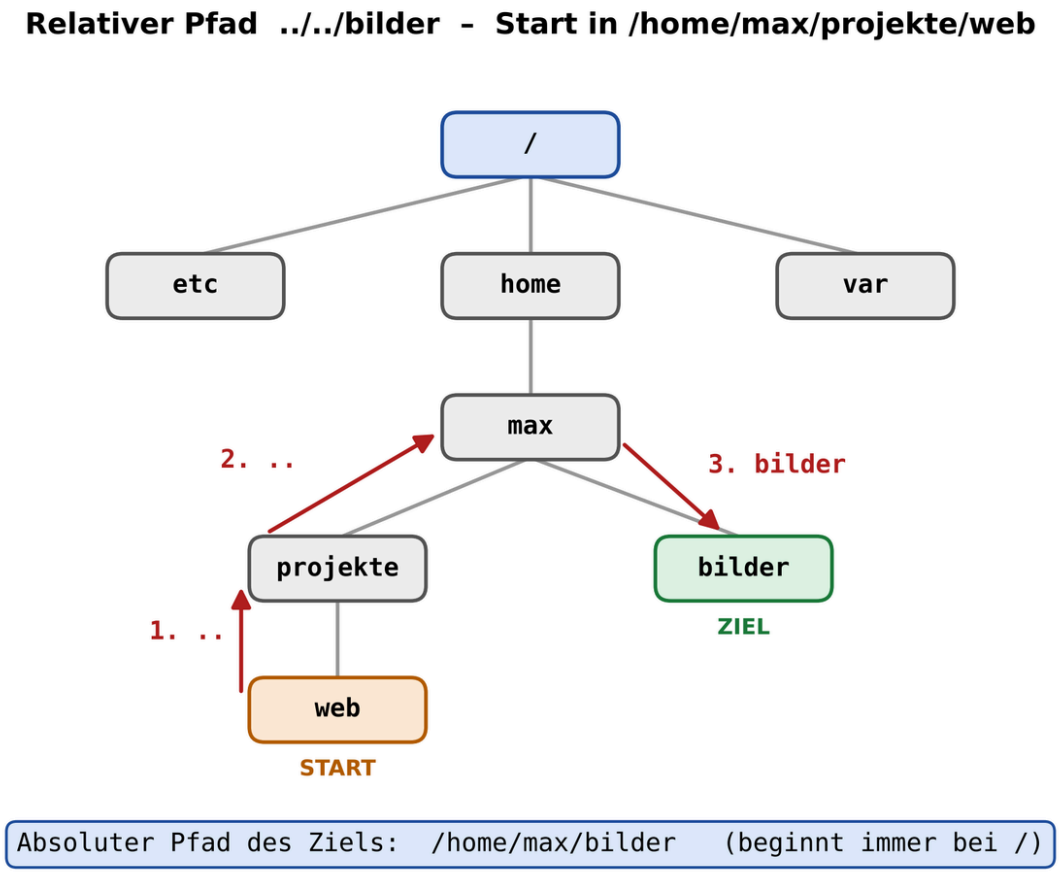
- Ein **absoluter Pfad** ist wie eine volle Postadresse. Er gilt immer, egal wo du stehst.
- Ein **relativer Pfad** ist wie “zwei Türen weiter links”. Er hängt davon ab, wo du gerade stehst.

Pfadart	Beginnt mit	Abhängig vom aktuellen Verzeichnis?	Beispiel
absoluter Pfad	`/`	nein, immer eindeutig	`/home/stephan/Musik/Obscura`
relativer Pfad	Name, `.` oder `..`	ja	`../../`

Ein relativer Pfad startet im aktuellen **Arbeitsverzeichnis**. ⚠️ Derselbe relative Pfad kann an anderer Stelle ein anderes Ziel haben.

🧱 Beispiel 6: relativen Pfad auflösen

Aktuelles Verzeichnis: `/home/max/projekte/web`. Befehl: `ls -l ../../bilder`



Verzeichnisbaum: von web zweimal nach oben bis max, dann hinunter nach bilder. Absoluter Pfad /home/max/bilder

Verzeichnisbaum: von web zweimal nach oben bis max, dann hinunter nach bilder. Absoluter Pfad /home/max/bilder

Schritt	Teil	Position danach
Start		`/home/max/projekte/web`
1	`..`	`/home/max/projekte`
2	`..`	`/home/max`
3	`bilder`	`/home/max/bilder`

Der absolute Pfad lautet also `/home/max/bilder`.

4.2 🏠 pwd und Tilde IHK gering

> 🎯 **IHK: gering** Grundbefehle, die man können muss, die aber selten einzeln geprüft werden

Zeichen / Befehl	Bedeutung
`pwd`	print working directory: zeigt den absoluten Pfad des aktuellen Verzeichnisses
`~`	Home-Verzeichnis des angemeldeten Benutzers

`ls -l ~` entspricht für stephan `ls -l /home/stephan`. ⚠️ Für Root zeigt `~` auf `/root`, nicht auf `/home/root`.

✏️ Übungsfragen Block 4

Ü4.1 Sie stehen in `/var/log/apache2`. Geben Sie den absoluten Pfad an, auf den `../../etc` zeigt.

Ü4.2 Root führt `cd ~` und danach `pwd` aus. Was wird ausgegeben, und warum?

🔧 Block 5: pipe, find, locate, cron

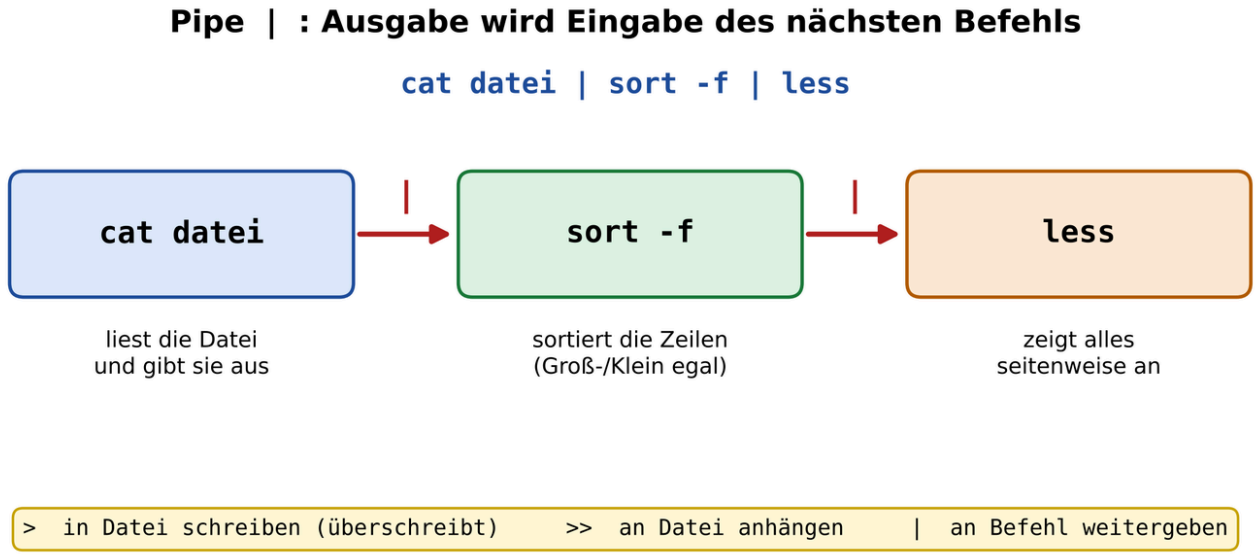
5.1 🚰 Umleitung und Pipe IHK hoch

> 🎯 **IHK: hoch** Befehlsketten sind Grundlage für Skripte und Automatisierung, die der Leitfaden für AP2 Konzeption/Administration ausdrücklich nennt

Stell dir ein Fließband vor. Jede Station macht einen Arbeitsschritt. Dann reicht sie das Teil weiter. Genau das macht die **Pipe** `|`. Sie gibt die Ausgabe eines Befehls an den nächsten Befehl weiter.

Zeichen	Name	Wirkung
`>`	Umleitung	Ausgabe in Datei schreiben, Datei wird überschrieben
`>>`	Umleitung	Ausgabe an Datei anhängen
`|`	**Pipe** (Verkettungszeichen)	Ausgabe eines Befehls wird Eingabe des nächsten

💡 Merksatz: `>` und `>>` = Umleitung in eine Datei. `|` = Verkettung von Befehlen.



Pipe-Ablauf: cat datei gibt den Inhalt an sort -f weiter, sort gibt die sortierten Zeilen an less weiter

Pipe-Ablauf: cat datei gibt den Inhalt an sort -f weiter, sort gibt die sortierten Zeilen an less weiter

Beispiel	Wirkung
`dmesg | less`	Kernel-Meldungen seitenweise und durchsuchbar anzeigen
`ls | wc -l`	Anzahl der Einträge im aktuellen Verzeichnis zählen
`cat datei | sort -f | less`	Datei sortiert (Groß-/Kleinschreibung ignoriert) anzeigen
`cat /etc/passwd | grep sp`	nur Zeilen mit “sp” anzeigen

`dmesg` (display messages) zeigt Kernel-Meldungen. Sie stammen aus dem Kernel-Ringpuffer.

Ohne Pipe bräuchte man eine Zwischendatei:

```
$ dmesg > tempfile
$ less tempfile
```

5.2 🔎 find IHK mittel

> 🎯 **IHK: mittel** Dateisuche nach Kriterien ist typische Administrationspraxis in AP2 Konzeption/Administration

`find` ist wie eine Suche mit der Taschenlampe. Es durchsucht das Dateisystem live ab einem Startpunkt. Syntax: `find <startpunkt> <kriterien>`

Kriterium	Bedeutung	Beispiel
`-name`	Name, Groß-/Kleinschreibung zählt	`find /etc -name interfaces`
`-iname`	Name, Groß-/Kleinschreibung egal	`find /etc -iname InTeRfAcEs`
`-type f` / `d` / `l`	reguläre Datei / Verzeichnis / Symlink	`find /etc -type f -name "*.conf"`
`-size`	Größe; `+` größer, `-` kleiner	`find /var -size +100M`
`-user` / `-group`	Besitzer / Gruppe	`find /home -user max`
`-mtime`	Änderungszeit in Tagen	`find /var/log -mtime -7` (in den letzten 7 Tagen geändert)
`-atime`	letzter Zugriff in Tagen	`find /home -atime +30` (länger als 30 Tage nicht gelesen)
`-perm`	Berechtigungen	`find /home -perm 644`

Man kann Kriterien kombinieren. Dann müssen alle zutreffen.

5.3 📇 locate IHK gering bis mittel

> 🎯 **IHK: gering bis mittel** der Unterschied find/locate (live vs. Datenbank) ist eine gute Abgrenzungsfrage, sonst Randthema

`locate` ist wie ein Telefonbuch. Es sucht nicht live. Es schaut in einer Liste nach, die vorher erstellt wurde. Das geht sehr schnell. Aber neue Einträge fehlen, bis die Liste erneuert wird.

	find	locate
Suche	live im Dateisystem	in einer vorab erzeugten Datenbank
Tempo	langsamer	sehr schnell
Kriterien	Name, Typ, Größe, Besitzer, Zeit, Rechte	nur Dateiname
Aktualität	immer aktuell	nur so aktuell wie die Datenbank

Beispiel: `locate resolv.conf`

Die Datenbank erneuert man mit Root-Rechten, z. B. `sudo updatedb` (Befehlsname nicht auf den Folien). Viele Systeme tun das automatisch per Cron-Job, etwa einmal täglich.

⚠️ Wenn locate nichts findet, gibt es zwei typische Ursachen:

1. Die Datenbank ist veraltet. Die Datei wurde nach der letzten Aktualisierung angelegt oder verschoben.
2. Die Datei existiert nicht unter diesem Namen. Oft ist es ein Tippfehler oder falsche Groß-/Kleinschreibung. Mit `locate -i` wird die Schreibweise ignoriert.

5.4 ⏰ Cron IHK mittel

> 🎯 **IHK: mittel** zeitgesteuerte Automatisierung passt zu “Programme zur automatisierten Systemverwaltung” in AP2 Konzeption/Administration

Ein **Cron-Job** ist wie ein Wecker für Aufgaben. Der Dienst cron führt die Aufgabe automatisch zu festen Zeiten aus. Der Name kommt von Chronos. Das ist griechisch und heißt Zeit.

Ergänzung, nicht auf den Folien: Cron-Jobs stehen in der **crontab**. Eine Zeile hat fünf Zeitfelder. Danach kommt der Befehl.

```
# Minute  Stunde  Tag  Monat  Wochentag  Befehl
  30      2       *    *      *          /usr/local/bin/backup.sh
```

Bedeutung: jeden Tag um 02:30 Uhr das Backup-Skript ausführen. `*` heißt “jeder Wert”. Bearbeiten mit `crontab -e`.

✏️ Übungsfragen Block 5

Ü5.1 Geben Sie einen Befehl an, der unter `/var/log` alle regulären Dateien mit Endung `.log` findet, die größer als 50 MB sind.

Ü5.2 Zählen Sie mit einer Befehlskette, wie viele Zeilen in `/etc/passwd` den Begriff `bash` enthalten.

Ü5.3 Sie legen `/home/max/notiz.txt` an. `locate notiz.txt` findet sie nicht. Nennen Sie die Ursache und die Lösung.

✅ Lösungen

Block 1

Ü1.1 Zwei von drei:

- feine Rechtevergabe pro Benutzer und Befehl;
- Protokoll, wer welchen Befehl ausgeführt hat;
- Root-Rechte nur kurz und bei Bedarf statt einer dauerhaften Root-Shell.

Außerdem muss bei sudo kein Root-Passwort weitergegeben werden.

Ü1.2 anna darf auf allen Hosts (`ALL`) den Befehl `/usr/bin/systemctl restart nginx` als root ausführen. Eine Passwortabfrage gibt es nicht (`NOPASSWD:`). `apt update` ist durch diese Regel nicht erlaubt. Ausnahme: anna hat über eine andere Regel bereits mehr Rechte, z. B. über die Gruppe sudo. sudoers-Regeln sind additiv.

Ü1.3 Ein Syntaxfehler kann sudo komplett blockieren. Dann kommt man ohne Root-Passwort nicht mehr an Admin-Rechte. Außerdem könnten zwei Personen gleichzeitig bearbeiten. Richtig ist `visudo` : Es sperrt die Datei und prüft die Syntax vor dem Speichern.

Block 2

Ü2.1 Oktal: Eigentümer `rwx` = 4+2+1 = 7, Gruppe `r-x` = 4+0+1 = 5, Andere `---` = 0. Ergebnis **750**. Die 2 ist die Anzahl der Hardlinks: Zwei Dateinamen zeigen auf denselben Inhalt. anna gehört zu “Andere” und hat `---` . Sie darf das Skript weder lesen noch ausführen.

Ü2.2

- Datei: Ausgangsrechte 666 = `rw-` `rw-` `rw-` .
  - Eigentümer, umask 0: nichts entfernen, `rw-` = 6.
  - Gruppe, umask 2 (`w`): `rw-` ohne `w` = `r--` = 4.
  - Andere, umask 7 (`rwx`): alles entfernen, `---` = 0.
  - Ergebnis: **640** = `rw-r-----` .
- Verzeichnis: Ausgangsrechte 777 = `rwx` `rwx` `rwx` .
  - Eigentümer: `rwx` = 7. Gruppe: `rwx` ohne `w` = `r-x` = 5. Andere: `---` = 0.
  - Ergebnis: **750** = `rwxr-x---` .

Ü2.3 Die Gruppe zeigt `s` (klein): SetGID ist gesetzt, und die Gruppe hat x. Andere zeigen `T` (groß): Sticky Bit ist gesetzt, aber Andere haben kein x.

- Grundrechte: `rwx` = 7, `rwx` = 7 (`s` zählt als x), `---` = 0 (`T` zählt nicht als x). Also 770.
- Spezialrechte: SetGID 2 + Sticky 1 = 3.
- Vollständiger Oktalwert: **3770**.

Block 3

Ü3.1 a) `/var/log` , b) `/etc` , c) `/root` , d) `/proc` .

Ü3.2 Die Aussage ist falsch. `/dev` ist ein virtuelles Verzeichnis. Der Kernel erzeugt es beim Systemstart laufend neu. Er stellt es im Arbeitsspeicher bereit. Die Gerätedateien bilden Hardware ab. Sie sind keine gespeicherten Dateien auf der Festplatte.

Ü3.3 usrmerge bedeutet: `/bin` und `/sbin` sind keine eigenen Verzeichnisse mehr. Sie sind Symlinks auf `/usr/bin` und `/usr/sbin` . Prüfen mit `ls -ld /bin /sbin` . Steht vorne ein `l` und ein Pfeil `->` , ist es ein Symlink.

Block 4

Ü4.1 Start `/var/log/apache2`. Das erste `..` führt zu `/var/log`. Das zweite `..` führt zu `/var`. Dann `etc`. Ergebnis: `/var/etc`. ⚠️ Achtung: Das ist nicht `/etc`. Ein relativer Pfad hängt immer vom Startpunkt ab.

Ü4.2 Ausgabe: `/root`. Die Tilde zeigt auf das Home-Verzeichnis des angemeldeten Benutzers. Bei Root liegt es standardmäßig in `/root`, nicht in `/home/root`.

Block 5

Ü5.1

```
$ find /var/log -type f -name "*.log" -size +50M
```

Für Verzeichnisse ohne Leserechte braucht man ggf. `sudo` davor.

Ü5.2

```
$ cat /etc/passwd | grep bash | wc -l
```

Kürzer geht es mit `grep bash /etc/passwd | wc -l` oder `grep -c bash /etc/passwd`.

Ü5.3 locate sucht in einer Datenbank, nicht live. Die Datei ist neuer als die letzte Aktualisierung. Lösung: Datenbank mit Root-Rechten erneuern (`sudo updatedb`). Oder live suchen mit `find /home/max -name notiz.txt`.

📝 Schlagwortliste für die Prüfung

Begriff	Kurzdefinition	Block
Root / Superuser	Konto mit allen Rechten auf dem System	1
su	Wechsel zu einem anderen Benutzer (substitute/switch user)	1
sudo	Einzelnen Befehl mit Root-Rechten ausführen, wird protokolliert	1
Gruppe sudo	Mitglieder dürfen unter Debian/Ubuntu alle Befehle per sudo ausführen	1
sudoers / visudo	Konfiguration der sudo-Rechte / sicherer Editor mit Sperre und Syntaxprüfung	1
NOPASSWD	sudoers-Option: keine Passwortabfrage	1

Begriff	Kurzdefinition	Block
Shell-Prompt	`$` Benutzer, `#` Root, `>` Fortsetzung	1
Multiuser-Betriebssystem	Mehrere Benutzer arbeiten parallel, Rechte trennen ihre Daten	2
r / w / x	Lesen 4, Schreiben 2, Ausführen 1	2
u / g / o / a	Eigentümer, Gruppe, Andere, alle	2
Oktalschreibweise	Rechte als drei Ziffern 0–7, z. B. 755	2
Symbolische Schreibweise	Rechte mit `+`, `-`, `=`, z. B. `u+x`	2
chmod / chown / chgrp	Rechte / Eigentümer / Gruppe ändern	2
Hardlink	weiterer Name für denselben Dateiinhalt, nur im selben Dateisystem	2
Symbolischer Link	Verweis auf einen Pfad, kann ins Leere zeigen	2
umask	Maske, die Rechte bei neuen Dateien und Verzeichnissen entfernt	2
SetUID (4, `u+s`)	Programm läuft mit den Rechten des Dateibesitzers, z. B. passwd	2
SetGID (2, `g+s`)	Neue Dateien im Verzeichnis erben dessen Gruppe	2
Sticky Bit (1, `+t`)	Nur eigene Dateien im gemeinsamen Verzeichnis löschbar, z. B. /tmp	2
`S` / `T`	Spezialrecht gesetzt, aber x an dieser Stelle fehlt	2
Principle of Least Privilege	So wenig Rechte wie möglich, so viele wie nötig	2
Zero Trust	Nichts automatisch vertrauen, jeden Zugriff ständig prüfen	2
Root-Verzeichnis `/`	einzige Wurzel des Linux-Verzeichnisbaums	3
FHS	Filesystem Hierarchy Standard: legt Standardverzeichnisse fest	3
`/etc`	systemweite Konfigurationsdateien	3
`/var/log`	Logdateien	3

Begriff	Kurzdefinition	Block
`/home` / `/root`	Home-Verzeichnisse der Benutzer / von Root	3
`/proc`, `/sys`, `/dev`	virtuelle Verzeichnisse, vom Kernel dynamisch erzeugt	3
usrmerge	`/bin` und `/sbin` sind Symlinks auf `/usr/bin` und `/usr/sbin`	3
tee	Eingabe ausgeben und gleichzeitig in Datei schreiben	3
Versteckte Datei	Name beginnt mit Punkt, Anzeige mit `ls -la`	3
`.` / `..`	aktuelles / übergeordnetes Verzeichnis	3
Absoluter Pfad	beginnt bei `/`, immer eindeutig	4
Relativer Pfad	beginnt im aktuellen Arbeitsverzeichnis	4
pwd	zeigt den absoluten Pfad des Arbeitsverzeichnisses	4
Tilde `~`	Home-Verzeichnis des angemeldeten Benutzers	4
Umleitung `>` / `>>`	Ausgabe in Datei schreiben / anhängen	5
Pipe `|`	Ausgabe eines Befehls als Eingabe des nächsten	5
find	Live-Suche nach Name, Typ, Größe, Besitzer, Zeit, Rechten	5
locate	schnelle Namenssuche in einer Datenbank	5
Cron-Job / crontab	zeitgesteuerte Aufgabe / Tabelle der Cron-Jobs	5

Zusammengeführt aus 3 Einzel-Zusammenfassung(en) · Stand 25.11.2026