Linux Modul 2 (Grundlagen II) – Zusammenfassung mit IHK-Bezug

Shell, Grundbefehle, Optionen, Hilfe, Bash-Tipps, Texteditoren, SSH – in einfacher Sprache

🐧 Linux Modul 2 – Zusammenfassung mit IHK-Bezug

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

📖 So liest du dieses Dokument

- 🎯 **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

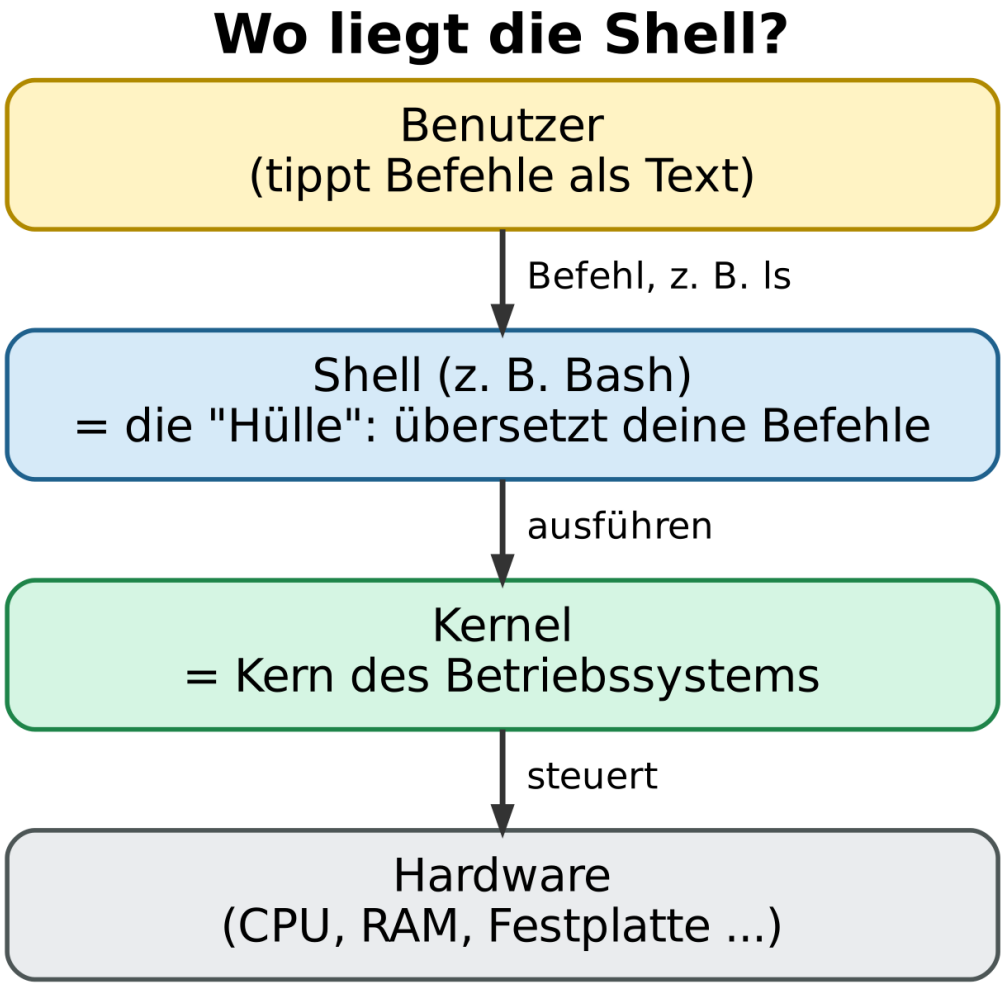
1.1 Was ist eine Shell?

📊 IHK-Relevanz: 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

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-Relevanz: 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;	Standard-Shell unter Debian

Shell	Name und Herkunft	Besonderheit
	Wortspiel auf die Bourne Shell (`sh`) von Stephen Bourne	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-Relevanz: 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).

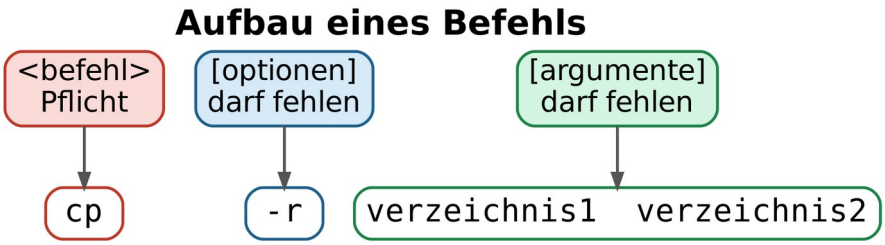
Zeichen	Bedeutung
`$`	normaler Benutzer
`#`	Root

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]
```




```
< ... > = Pflicht: Platzhalter durch echten Wert ersetzen
[ ... ] = optional: darf fehlen
```

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-Relevanz: 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,

Befehl	Herkunft	Zweck
		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-Relevanz: 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:

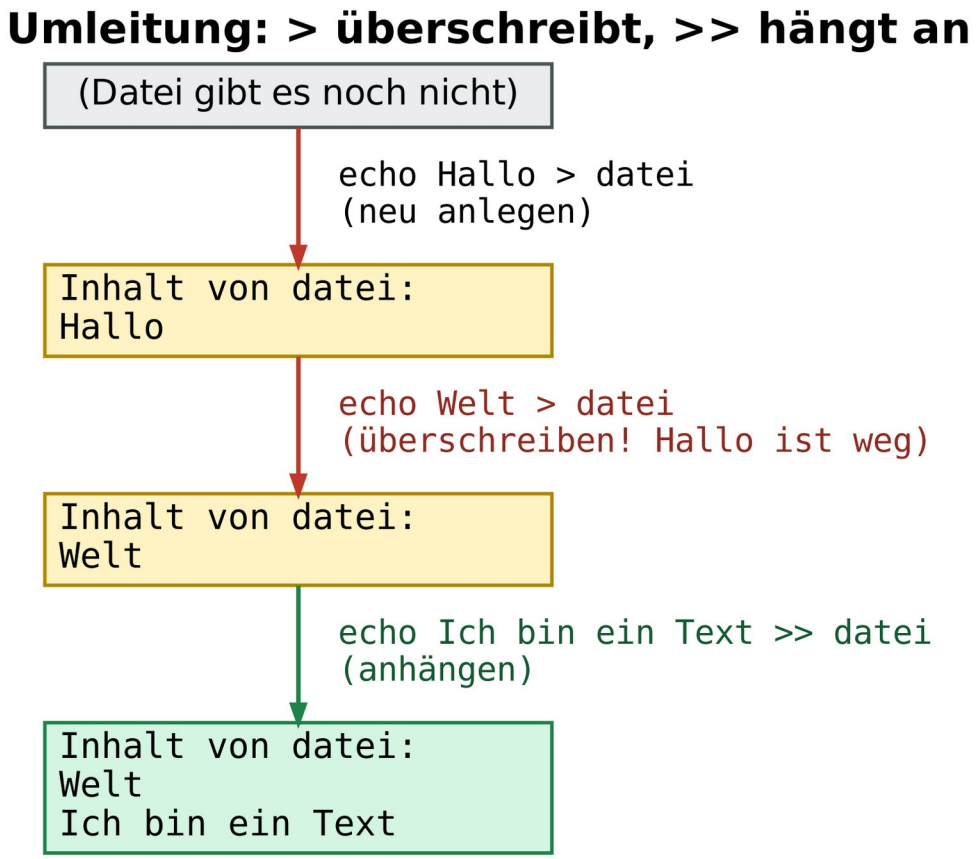
- `>` ist wie ein Blatt **neu beschreiben**. Was vorher draufstand, ist weg.
- `>>` ist wie eine Zeile **unten dazuschreiben**. Das Alte bleibt stehen.

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 >>

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-Relevanz: 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-Relevanz: 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-Relevanz: 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-Relevanz: 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.

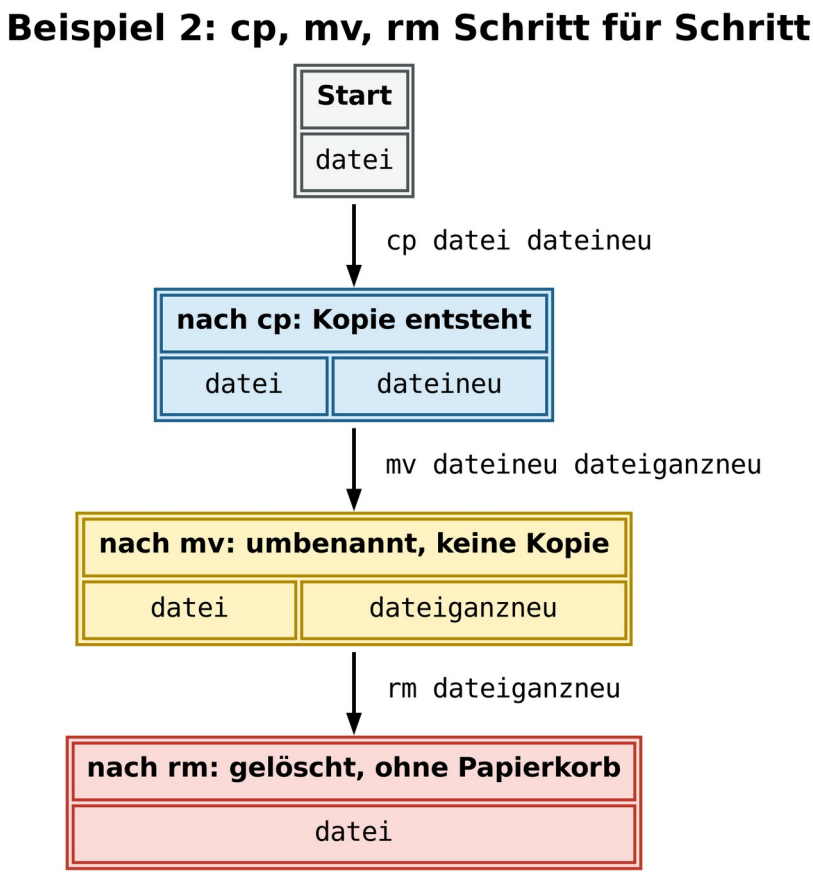
Befehl	Syntax	Wirkung
`cp`	`cp <quelle> <ziel>`	Kopie anlegen, Original bleibt
`cp -r`	`cp -r <verz> <ziel>`	Verzeichnis samt Inhalt kopieren (rekursiv)
`mv`	`mv <quelle> <ziel>`	verschieben, es entsteht keine Kopie
`mv`	`mv alt.txt neu.txt`	umbenennen
`rm`	`rm <datei>`	Datei löschen
`rm -r`	`rm -r <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

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.

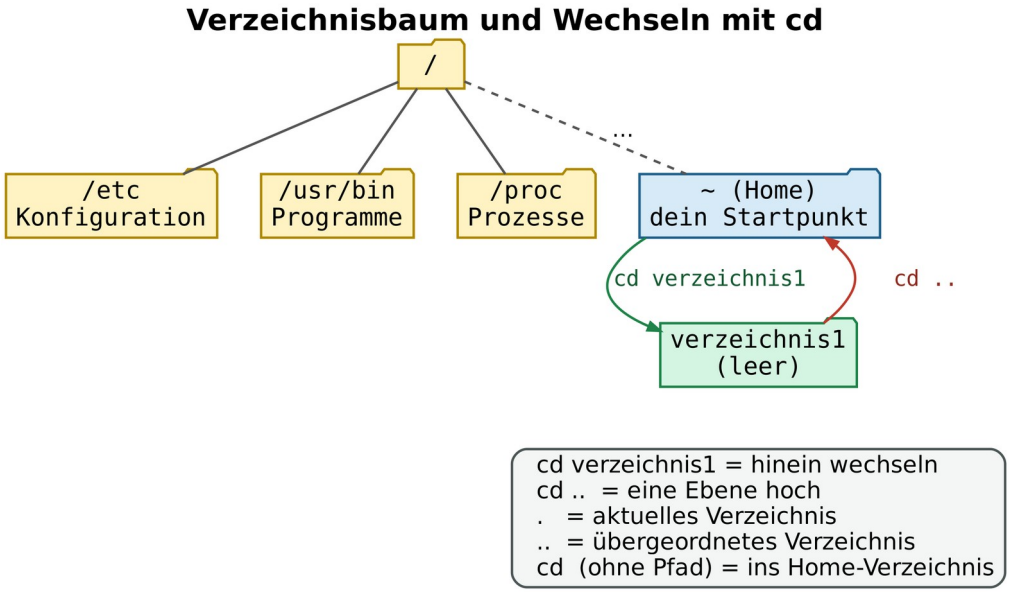
2.7 🗂️ mkdir, rmdir, cd – mit Verzeichnissen arbeiten

📊 IHK-Relevanz: 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

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-Relevanz: 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`
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

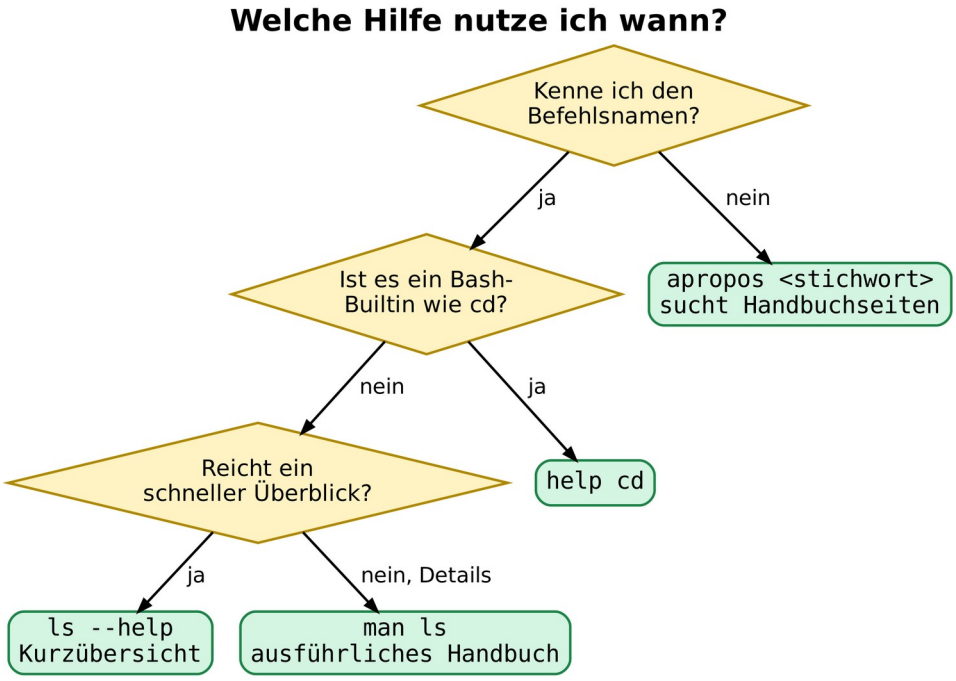
📊 IHK-Relevanz: mittel – “Wie helfe ich mir selbst?” ist eine typische Praxisfrage. In AP1 passt sie zu Betriebssystemen und Kundeneinweisung.

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

🔍 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-Relevanz: 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-Relevanz: 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-Relevanz: 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-Relevanz: 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-Relevanz: gering – Praxiswissen. Ein Prüfer erwartet eher, dass man einen Editor sicher bedienen kann.

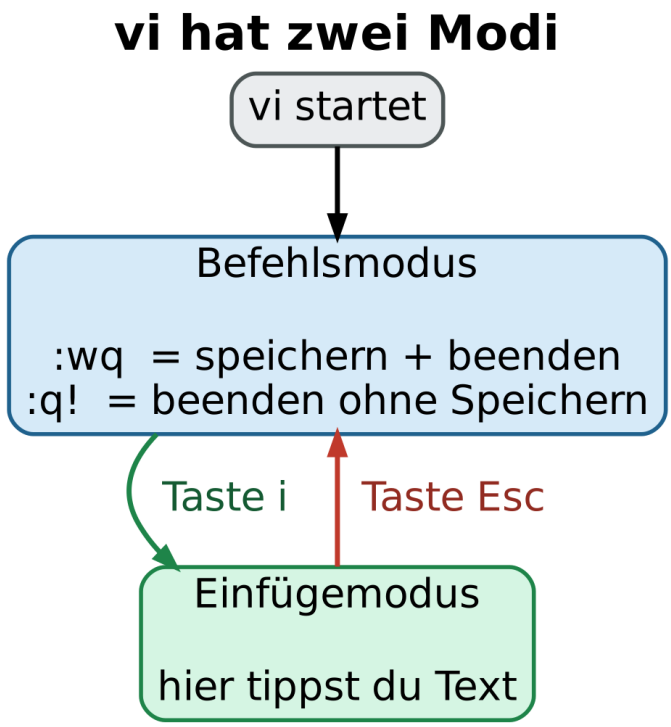
Editor	Speichern und beenden	Ohne Speichern beenden
vi / vim	Esc, dann `:wq`, Enter	Esc, dann `:q!`, Enter
nano	Strg + O, Enter, dann Strg + X	Strg + X, dann N
emacs	Strg + X, Strg + S; dann Strg + X, Strg + C	Strg + X, Strg + C; Speichern mit `n` ablehnen, mit `yes` bestätigen
JOE	Strg + K, dann X	Strg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

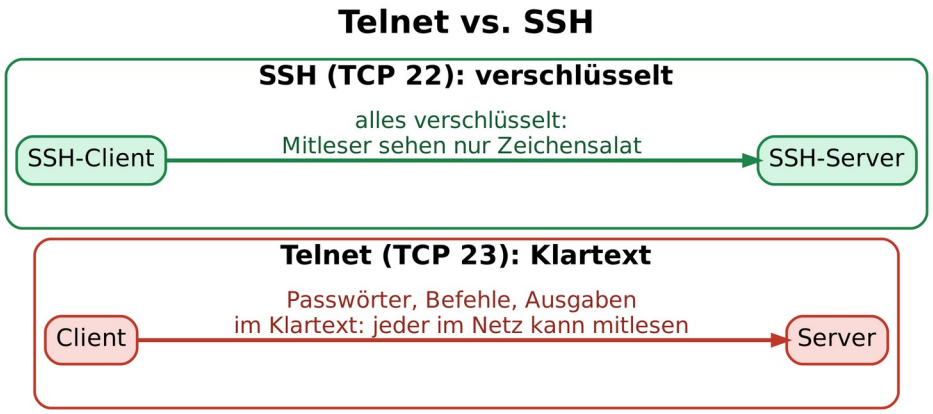
6.1 Was ist SSH?

📊 IHK-Relevanz: 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

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-Relevanz: 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-Relevanz: mittel – Dienste installieren, starten und aktivieren ist Administrationspraxis (AP2 Konzeption/Administration).

Auf dem Zielsystem (als Root):

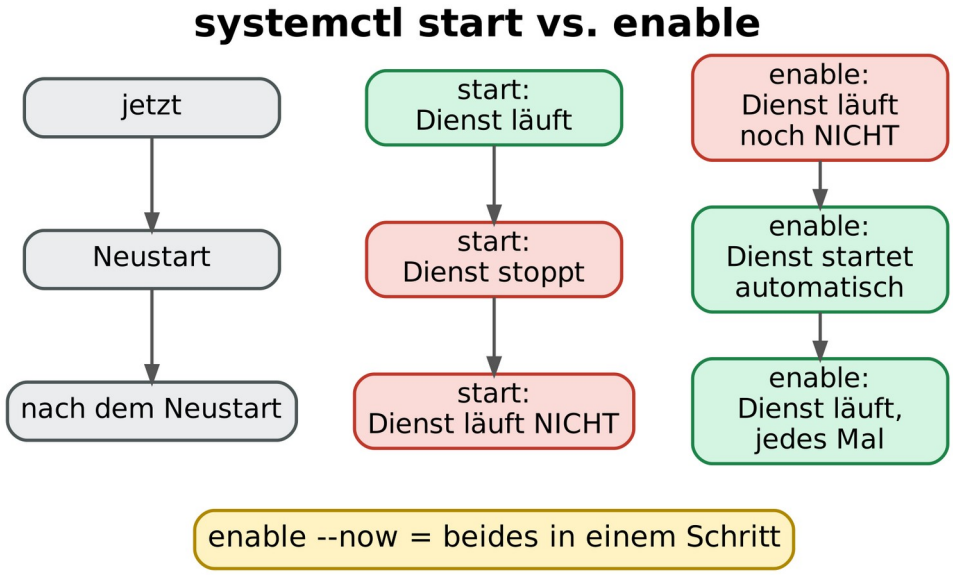
```
# apt update
# apt install openssh-server
# systemctl start ssh
```

```
# 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

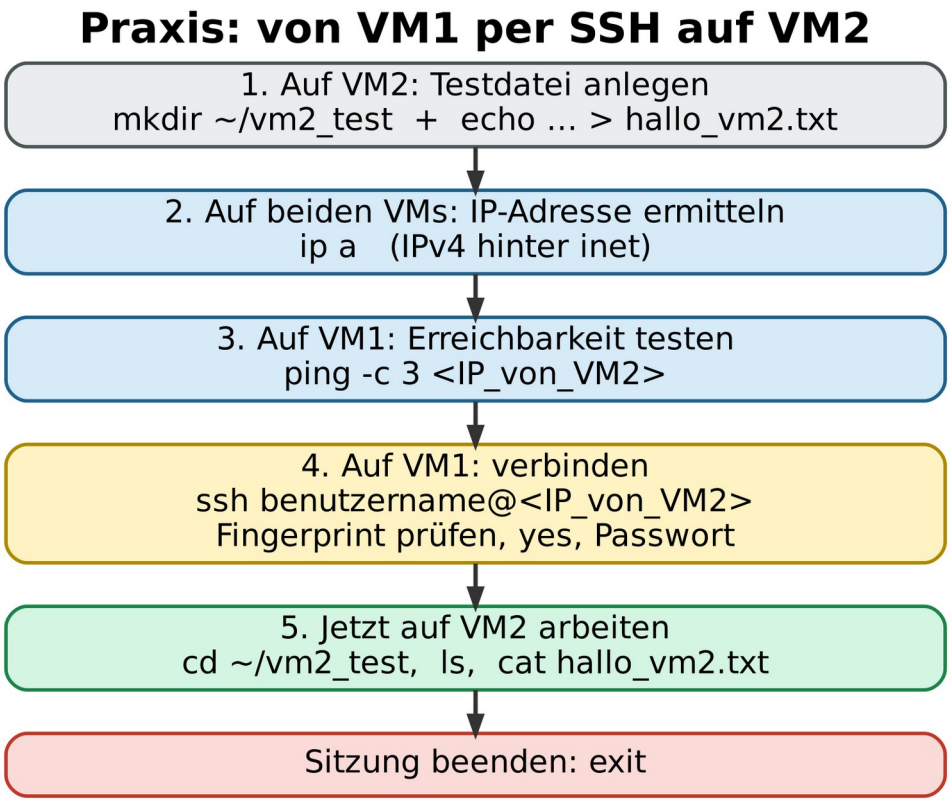
> 💡 **Merksatz:** `start` wirkt sofort, aber nur bis zum Neustart. `enable` wirkt erst ab dem nächsten Start, dann aber dauerhaft.

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

6.4 🖧 Praxis: Von VM1 per SSH auf VM2

📊 IHK-Relevanz: 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

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
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	2

Begriff	Kurzdefinition	Block
	Verzeichnis löschen	
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
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-	6

Begriff	Kurzdefinition	Block
	Fingerprints	
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