Linux Modul 3 (Grundlagen III) – Zusammenfassung mit IHK-Bezug

Root und sudo, Dateirechte, Verzeichnisse, Pfade, pipe/find/locate/cron

🐧 Linux Modul 3 – Zusammenfassung mit IHK-Bezug

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

🧭 So liest du dieses Dokument

- **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.

🔣 Bedeutung der Symbole

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-Relevanz: 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**.

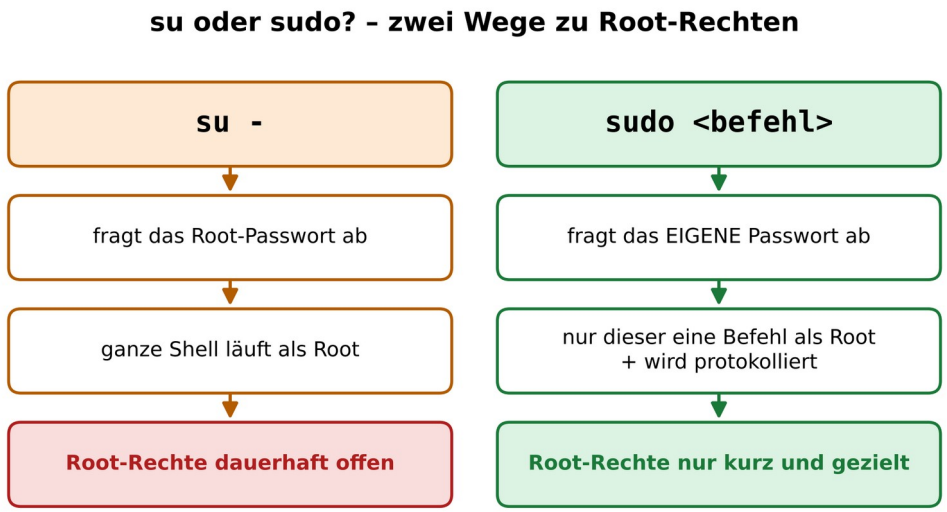
Root darf alles. Root darf jede Datei lesen, ändern und löschen. Unter Windows ist das wie das eingebaute Administrator-Konto.

⚠️ 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

📖 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-Relevanz: 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` =

Befehl	Wirkung
	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-Relevanz: 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-Relevanz: 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

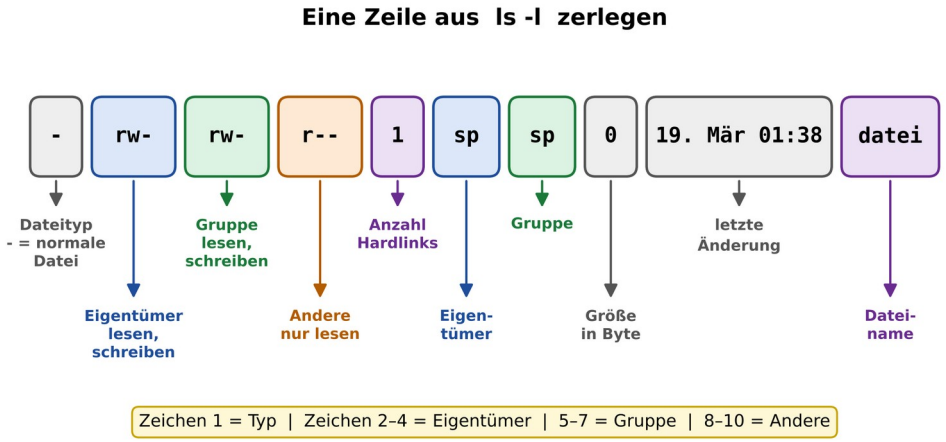
Die Rechte gelten für drei Klassen:

- 👤 **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

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`

Zeichen	Typ
`b`	Block-Gerätedatei (block device), z. B. Festplatten
`s`	Socket
`p`	Named Pipe (FIFO): Datenaustausch zwischen zwei Prozessen

2.2 🔗 Hardlinks und symbolische Links

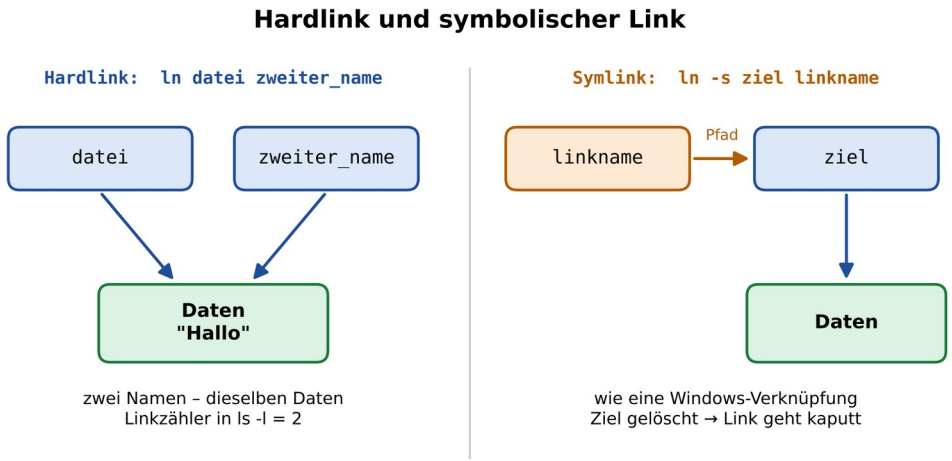
🎯 IHK-Relevanz: mittel – Unterschied Hardlink/Symlink ist ein klassisches Abgrenzungsthema für AP2 Konzeption/Administration.

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

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	bleibt gültig, wenn ein anderer Name

Symbolischer Link (Symlink)	Hardlink
verschoben wird	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-Relevanz: 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

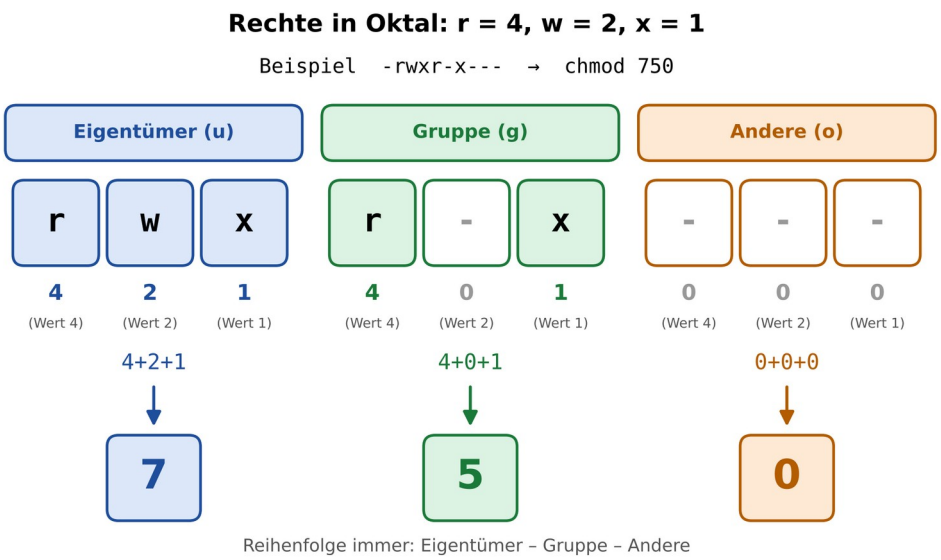
Jedes Recht hat einen Zahlenwert: **r = 4, w = 2, x = 1**. Pro Klasse zählt man die Werte zusammen. So entsteht eine Ziffer von 0 bis 7. Das ist wie Münzen zählen: Eine 4er-, eine 2er- und eine 1er-Münze.

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

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-Relevanz: 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-Relevanz: 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`

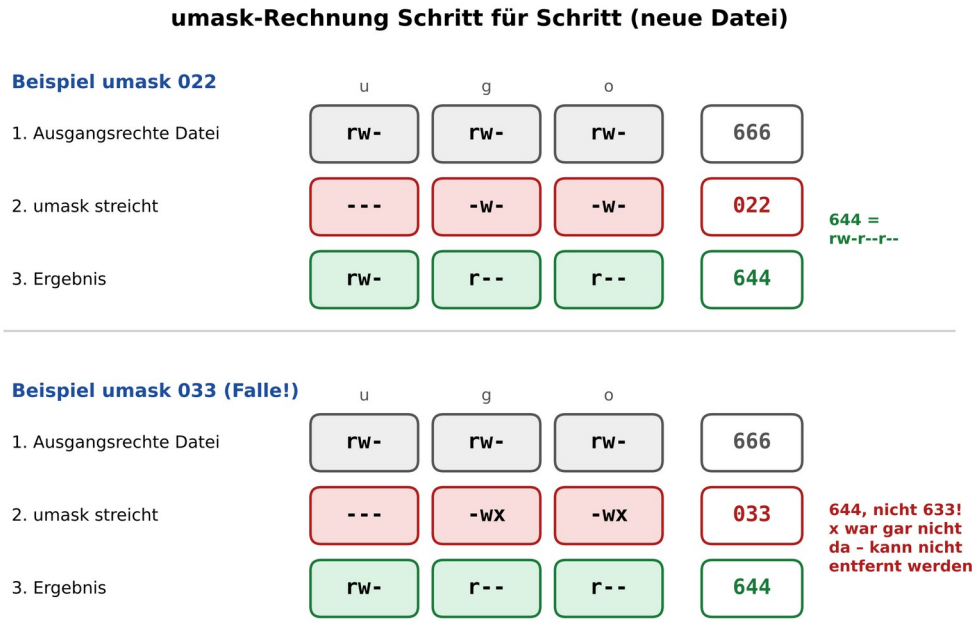
⚠️ **Wichtiger Hinweis zur Rechnung:** Die Folien rechnen “666 − 022 = 644”. Das klappt bei den üblichen Werten. Genau genommen **entfernt** die umask aber Rechte-Bits. Sie zieht nicht Stelle für Stelle ab. Ein Recht, das gar nicht da ist, kann man nicht entfernen.

🧮 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

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	644 `rw-r--r--`	755 `rwxr-xr-x`

umask	Bedeutung	Neue Dateien	Neue Verzeichnisse
	Anderen entfernen		
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------`

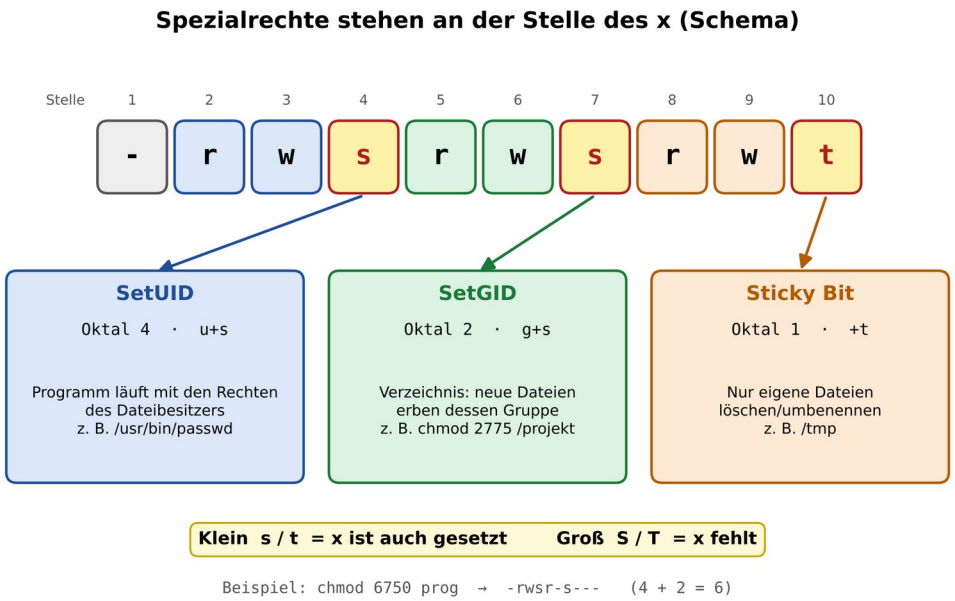
Befehl	Wirkung
`umask`	aktuelle Maske anzeigen
`umask 027`	neue Maske setzen, gilt nur für die aktuelle Shell-Sitzung
`umask -S`	Maske symbolisch anzeigen; bei 022: `u=rwx,g=rx,o=rx`

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-Relevanz: 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

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	x ist gesetzt

Anzeige	Spezialrecht	x-Recht an dieser Stelle
	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-Relevanz: 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-Relevanz: 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-Relevanz: 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)
`/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-Relevanz: 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-Relevanz: mittel – `.` und `..` sind Voraussetzung für Pfadaufgaben; versteckte Dateien sind Basiswissen.

Eintrag	Bedeutung
`.`	aktuelles Verzeichnis
`..`	übergeordnetes Verzeichnis
`.name`	**versteckte Datei** bzw. verstecktes

Eintrag	Bedeutung
	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-Relevanz: mittel – Pfadangaben tauchen in jeder Shell- oder Skriptaufgabe auf (AP2 Konzeption/Administration), selten als eigene Frage.

Ein Pfad ist wie eine Wegbeschreibung.

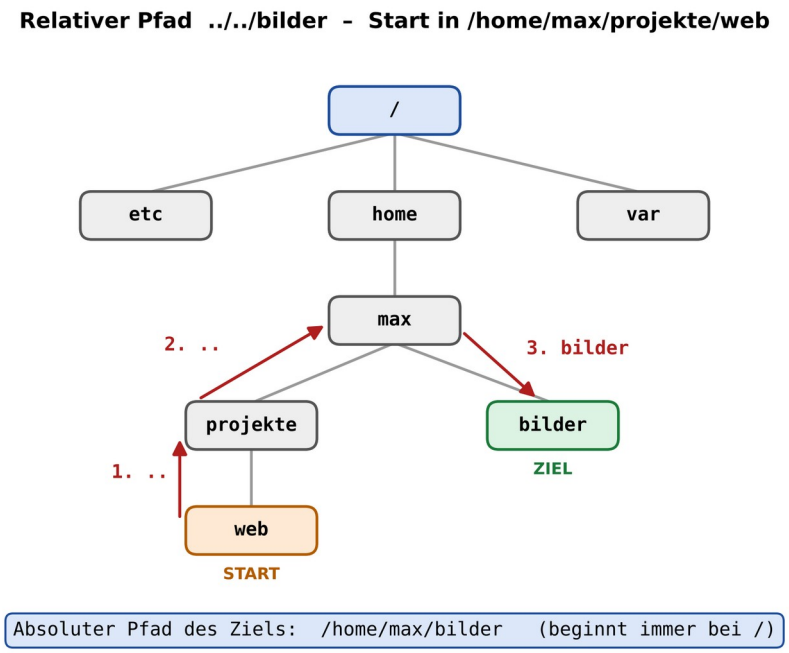
- Ein **absoluter Pfad** ist wie eine volle Postadresse. Er gilt immer, egal wo du stehst.
- Ein **relativer Pfad** ist wie "zwei Türen weiter links". Er hängt davon ab, wo du gerade stehst.

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

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-Relevanz: 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

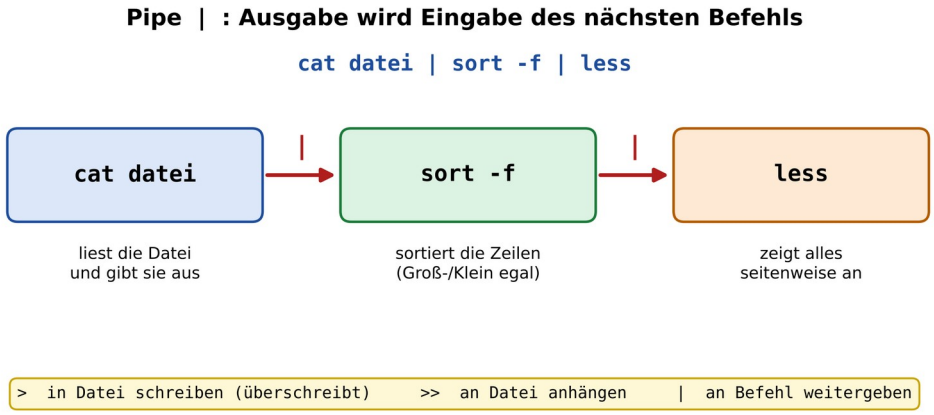
5.1 🚰 Umleitung und Pipe

🎯 IHK-Relevanz: 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

Beispiel	Wirkung
`dmesg | less`	Kernel-Meldungen seitenweise und durchsuchbar anzeigen

Beispiel	Wirkung
`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-Relevanz: 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-Relevanz: 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-Relevanz: 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-	1

Begriff	Kurzdefinition	Block
	Rechten ausführen, wird protokolliert	
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
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	2

Begriff	Kurzdefinition	Block
	an dieser Stelle fehlt	
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
`/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

Begriff	Kurzdefinition	Block
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