Inhaltsverzeichnis

id

Beschreibung: Gibt die UID (User ID), die primäre GID (Group ID) sowie alle Gruppen des aktuellen oder eines angegebenen Benutzers aus. Besonders nützlich zur schnellen Überprüfung von Rechten im System.

Syntax: id [OPTIONEN] [BENUTZER]

UID-Bereiche (Linux-Konvention)

Bereich	Bedeutung
0	root – Superuser mit vollen Systemrechten
1 – 999	System-/Dienst-Konten (daemon, www-data, ...)
1000 – 65533	Normale Benutzer
65534	nobody – minimale Rechte, für NFS o. ä.

Flags

Flag	Langform	Bedeutung
-u	--user	Nur die UID (User ID) numerisch ausgeben
-g	--group	Nur die primäre GID numerisch ausgeben
-G	--groups	Alle GIDs (primär + alle sekundären) aufzählen
-n	--name	Namen statt numerischer ID ausgeben – immer mit -u, -g oder -G kombinieren
-r	--real	Echte (reale) ID statt der effektiven ID ausgeben
-Z	--context	SELinux-Sicherheitskontext ausgeben (nur auf SELinux-Systemen verfügbar)

Beispiele

```
# Alle IDs des aktuell eingeloggten Benutzers ausgeben
id

# Nur die eigene UID numerisch
id -u

# Nur die eigene UID als Namen (Benutzername)
id -un

# Alle Gruppen-IDs des aktuellen Nutzers
id -G

# Alle Gruppennamen (keine Nummern)
id -Gn

# IDs eines anderen Benutzers abfragen (als root oder für eigenen Account)
id www-data

# IDs des Benutzers "alice" prüfen – nützlich zur Berechtigungsprüfung
id alice

# Prüfen ob ein Benutzer in einer bestimmten Gruppe ist (z.B. sudo)
id -Gn alice | grep -w sudo
```

Ausgabe

```
# id
uid=1000(matta) gid=1000(matta)
Gruppen=1000(matta),4(adm),27(sudo),999(docker)

# id -u
1000

# id -un
matta

# id -Gn
matta adm sudo docker

# id www-data
uid=33(www-data) gid=33(www-data) Gruppen=33(www-data)

# id -Gn alice | grep -w sudo
sudo          <- alice ist in der sudo-Gruppe
```

Hinweis: *Mit **id** lässt sich schnell überprüfen ob ein Benutzer Sudo-Rechte hat (**sudo** in Gruppen) oder zu Dienst-Gruppen wie **docker** gehört.*

umask

Beschreibung: umask (User file-creation Mode Mask) bestimmt, welche Berechtigungsbits beim Erstellen neuer Dateien und Verzeichnisse **automatisch entfernt** werden. Sie gilt für die aktuelle Shell-Sitzung und alle daraus gestarteten Prozesse.

Syntax: umask [OPTIONEN] [MASKE]

Funktionsprinzip

Die umask wirkt als **Subtraktion** von den maximalen Standardrechten:

	Datei	Verzeichnis
Maximale Rechte:	666	777
minus umask:	-022	-022
	---	---
Ergebnis:	644	755

Wichtig: *Bei Dateien ist das Ausführ-Bit (1) grundsätzlich nicht im Maximum enthalten. Dateien werden nie direkt als ausführbar erstellt.*

Flags

Flag	Bedeutung
-S	Symbolische Darstellung ausgeben (z. B. u=rwx,g=rx,o=rx)
-p	Aktuelle umask als umask-Befehl ausgeben (für Shell-Skripte / Sicherung geeignet)

Häufige umask-Werte

umask	Dateien	Verzeichnisse	Typischer Einsatz
0022	rw-r--r-- (644)	rwxr-xr-x (755)	Standard auf den meisten Systemen
0027	rw-r----- (640)	rwxr-x--- (750)	Sicherere Server-Umgebungen
0077	rw------- (600)	rwx------ (700)	Sehr restriktiv (nur Eigentümer)
0002	rw-rw-r-- (664)	rwxrwxr-x (775)	Kollaborative Umgebungen

Beispiele

```
# Aktuelle umask numerisch anzeigen
umask

# Aktuelle umask symbolisch anzeigen
umask -S

# umask auf 027 setzen – gilt ab diesem Moment für alle neuen Dateien
umask 027

# So prüfen welche Dateirechte mit der aktuellen umask entstehen
umask 022
touch testdatei.txt
ls -l testdatei.txt

# umask dauerhaft setzen: in ~/.bashrc oder /etc/profile eintragen
echo "umask 027" >> ~/.bashrc

# umask als Befehl ausgeben (Skript-Sicherung)
umask -p

# Temporäre umask in einem Subshell-Block (nur in dieser Subshell aktiv)
(umask 077; touch geheim.txt; ls -l geheim.txt)
```

Ausgabe

```
# umask
0022

# umask -S
u=rwx,g=rx,o=rx

# umask 027 ; umask -S
u=rwx,g=rx,o=

# umask -p
umask 0022

# touch testdatei.txt ; ls -l testdatei.txt   (bei umask 022)
-rw-r--r-- 1 matta matta 0 Apr 29 10:00 testdatei.txt

# (umask 077; touch geheim.txt; ls -l geheim.txt)
-rw------- 1 matta matta 0 Apr 29 10:00 geheim.txt
```

Tipp: *Die umask gilt nur für die aktuelle Shell-Sitzung. Soll sie dauerhaft aktiv sein, muss sie in* ***/etc/profile*** *(systemweit) oder* ***~/.bashrc*** */* ***~/.profile*** *(benutzerspezifisch) eingetragen werden.*

chown

Beschreibung: Ändert den Eigentümer (**user**) und/oder die Gruppe (**group**) einer Datei oder eines Verzeichnisses. Nur **root** darf beliebige Eigentümer setzen; normale Benutzer können nur Dateien ihrer eigenen Gruppe zuweisen.

Syntax: chown [OPTIONEN] [EIGENTÜMER][:GRUPPE] DATEI...

Syntax-Varianten

Syntax	Wirkung
chown alice datei	Nur Eigentümer auf alice setzen
chown alice:dev datei	Eigentümer auf alice, Gruppe auf dev setzen
chown alice: datei	Eigentümer auf alice, Gruppe auf alice's Primärgruppe
chown :dev datei	Nur Gruppe ändern (Eigentümer unverändert)
chown 1001:1002 datei	Per numerischer UID:GID setzen

Flags

Flag	Bedeutung
-R	Rekursiv: Verzeichnis und alle enthaltenen Dateien/Unterverzeichnisse
-v	Verbose – zeigt jede durchgeführte Änderung an
-c	Wie -v, aber zeigt nur Zeilen mit tatsächlichen Änderungen
-f	Fehlermeldungen unterdrücken (force silent)
-h	Bei symbolischen Links: den Link selbst ändern, nicht das Ziel
--reference=REF	Eigentümer und Gruppe von der Datei REF übernehmen
--from=EIGEN:GRP	Nur Dateien ändern die momentan exakt diesen Eigentümer/Gruppe haben
-L	Bei -R: symbolische Links in Verzeichnisse verfolgen
-P	Bei -R: keine symbolischen Links verfolgen (Standard)

Beispiele

```
# Eigentümer einer einzelnen Datei ändern
chown alice datei.txt

# Eigentümer und Gruppe gleichzeitig setzen
chown alice:entwickler datei.txt

# Nur die Gruppe einer Datei ändern
chown :webteam /var/www/html/index.html

# Rekursiv für ein Webserver-Verzeichnis
chown -R www-data:www-data /var/www/html/

# Verbose: Jede Änderung protokollieren
chown -Rv matta:matta /home/matta/

# Eigentümer nur ändern wenn aktuell root gehört
chown --from=root:root alice:alice /home/alice/wichtig.txt

# Eigentümer/Gruppe von einer anderen Datei übernehmen
chown --reference=/etc/passwd /tmp/neue_passwd_kopie

# Symbolischen Link selbst (nicht Ziel) ändern
chown -h alice symlink_auf_datei

# Numerische UID und GID verwenden
chown 1000:1000 datei.txt
```

Ausgabe

```
# chown -v alice:alice /home/alice/test.txt
Eigentümer von '/home/alice/test.txt' von root:root zu alice:alice
geändert

# chown -Vc www-data:www-data /var/www/html/
Eigentümer von '/var/www/html/' von matta:matta zu www-data:www-data
geändert
Eigentümer von '/var/www/html/index.html' von matta:matta zu www-
data:www-data geändert
Eigentümer von '/var/www/html/style.css' von matta:matta zu www-
data:www-data geändert
```

Sicherheitshinweis: *Sei bei rekursivem* ***chown -R*** *auf* ***/*** *oder wichtigen Systemverzeichnissen äußerst vorsichtig – ein falsches Ziel kann das System unbrauchbar machen.*

chmod

Beschreibung: Ändert die Zugriffsrechte (Berechtigungen) von Dateien und Verzeichnissen. Die Rechte werden für drei Klassen vergeben: **Eigentümer (u)**, **Gruppe (g)** und **Andere (o)**.

Syntax: chmod [OPTIONEN] MODUS DATEI...

Bedeutung der Rechte

Recht	Zeichen	Oktal	Auf Dateien	Auf Verzeichnisse
Lesen	r	4	Dateiinhalt lesen	Verzeichnisinhalt auflisten (ls)
Schreiben	w	2	Datei bearbeiten/löschen	Dateien erstellen/löschen im Verz.
Ausführen	x	1	Datei ausführen	In Verzeichnis wechseln (cd)

Oktalnotation – vollständige Tabelle

Oktal	Binär	Symbolisch	Bedeutung
7	111	rwx	Lesen, Schreiben, Ausführen
6	110	rw-	Lesen, Schreiben
5	101	r-x	Lesen, Ausführen
4	100	r--	Nur Lesen
3	011	-wx	Schreiben, Ausführen
2	010	-w-	Nur Schreiben
1	001	--x	Nur Ausführen
0	000	---	Keine Rechte

Symbolische Notation

Zeichen	Bedeutung
u	Eigentümer (user/owner)
g	Gruppe (group)
o	Andere (others)
a	Alle (all = u+g+o)
+	Recht hinzufügen
-	Recht entfernen
=	Recht exakt setzen (andere Bits werden gelöscht)
r	Lesen (read = 4)
w	Schreiben (write = 2)
x	Ausführen (execute = 1)
X	Ausführen nur setzen wenn bereits woanders x gesetzt ist
s	SUID / SGID setzen
t	Sticky Bit setzen

Flags

Flag	Bedeutung
-R	Rekursiv – alle Dateien und Unterverzeichnisse
-v	Verbose – jede Änderung ausgeben
-c	Nur tatsächliche Änderungen ausgeben
-f	Fehlermeldungen unterdrücken
--reference=DATEI	Rechte von Referenzdatei übernehmen

Typische Oktalwerte

Oktal	Symbolisch	Verwendung
644	rw-r--r--	Standard für normale Dateien
755	rwxr-xr-x	Standard für Verzeichnisse/Skripte
600	rw-------	Private Schlüssel (.ssh/id_rsa)
700	rwx------	Private Verzeichnisse
664	rw-rw-r--	Gemeinsame Projektdateien
777	rwxrwxrwx	Alle Rechte für alle (gefährlich!)
640	rw-r-----	Sensible Konfigdateien

Beispiele

```
# Oktal: typisch für Shell-Skripte
chmod 755 skript.sh

# Oktal: typisch für normale Dateien
chmod 644 README.txt

# Oktal: nur Owner darf lesen und schreiben (z.B. SSH-Key)
chmod 600 ~/.ssh/id_rsa

# Symbolisch: Ausführrecht für alle hinzufügen
chmod a+x skript.sh

# Symbolisch: Schreibrecht für Andere entfernen
chmod o-w datei.txt

# Symbolisch: Alle Rechte exakt setzen (löscht alles was nicht
angegeben)
chmod u=rwx,g=rx,o= private.sh

# Symbolisch: Gruppe Schreibrecht hinzufügen
chmod g+w projekt/

# Rekursiv für Webverzeichnis
chmod -R 755 /var/www/html/

# Nur Ausführrecht setzen wenn es bereits für jemanden gesetzt war
chmod -R a+X verzeichnis/

# Rechte von anderer Datei übernehmen
chmod --reference=/etc/passwd /tmp/meine_passwdkopie

# Verbose anzeigen was geändert wird
chmod -vc 644 *.txt
```

Ausgabe

```
# Vorher (nach touch skript.sh)
-rw-r--r-- 1 matta matta 0 Apr 29 10:00 skript.sh

# chmod 755 skript.sh
-rwxr-xr-x 1 matta matta 0 Apr 29 10:00 skript.sh

# chmod 600 ~/.ssh/id_rsa
-rw------- 1 matta matta 2610 Apr 29 10:00 /home/matta/.ssh/id_rsa

# chmod -vc 644 *.txt
Modus von 'notizen.txt' geändert: 0664 (rw-rw-r--) -> 0644 (rw-r--r--)
```

Spezialbits

Beschreibung: Neben den normalen **rwx**-Bits existieren drei zusätzliche Sonderbits: **SUID**, **SGID** und das **Sticky Bit**. Sie werden im führenden vierten Oktalblock angegeben.

SUID (Set User ID) – Oktal: 4xxx

Auf ausführbare Dateien:

- Das Programm läuft mit den Berechtigungen des **Eigentümers** der Datei – nicht mit denen des ausführenden Benutzers.
- Typisches Beispiel: **/usr/bin/passwd** gehört **root**. Jeder Benutzer darf es ausführen und dadurch sein Passwort in **/etc/shadow** ändern.

Anzeige in ls -l:

- **s** statt **x** beim Eigentümer-Ausführbit, wenn SUID+x gesetzt: **-rwsr-xr-x**
- **S** (Großbuchstabe) wenn SUID gesetzt aber **kein** Ausführrecht: **-rwSr--r--**

```
# SUID symbolisch setzen
chmod u+s /usr/bin/myprog

# SUID oktal setzen (4 + normale Rechte)
chmod 4755 /usr/bin/myprog

# SUID entfernen
chmod u-s /usr/bin/myprog

# Alle SUID-Dateien im System finden (Sicherheitscheck!)
find / -type f -perm -4000 -ls 2>/dev/null

# Beispiel: passwd hat SUID
ls -l /usr/bin/passwd-rwsr-xr-x 1 root root 63960 Apr 13 2023
/usr/bin/passwd
   ^
   s = SUID + Ausführrecht
```

SGID (Set Group ID) – Oktal: 2xxx

Auf ausführbare Dateien:

- Das Programm läuft mit den Rechten der **Gruppe des Eigentümers** (nicht der Gruppe des Aufrufers).

Auf Verzeichnisse (häufigster Einsatz):

- Alle **neu erstellten Dateien und Unterverzeichnisse erben die Gruppe** des übergeordneten Verzeichnisses.
- Ideal für Teamverzeichnisse.

Anzeige in ls -l:

- **s** statt **x** beim Gruppen-Ausführbit: **drwxrwsr-x**
- **S** wenn SGID gesetzt aber kein Gruppen-Ausführrecht: **drwxrwSr-x**

```
# SGID auf einem gemeinsamen Projektverzeichnis setzen
chmod g+s /shared/projekt
chmod 2775 /shared/projekt

# Prüfen ob SGID gesetzt ist
ls -ld /shared/projekt

# Alle SGID-Verzeichnisse finden
find / -type d -perm -2000 -ls 2>/dev/nulldrwxrwsr-x 2 alice
entwickler 4096 Apr 29 10:00 /shared/projekt
      ^
      s = SGID auf Verzeichnis

# Neue Datei im Verzeichnis erstellt von Benutzer "bob":
touch /shared/projekt/bob_datei.txt
ls -l /shared/projekt/bob_datei.txt
-rw-r--r-- 1 bob entwickler 0 Apr 29 10:01 bob_datei.txt
               ^^^^^^^^^^ Gruppe wird vererbt!
```

Sticky Bit – Oktal: 1xxx

Auf Verzeichnisse:

- Dateien dürfen nur von ihrem **eigenen Eigentümer** oder **root** gelöscht/umbenannt werden – selbst wenn andere Benutzer Schreibrecht auf das Verzeichnis haben.
- Klassisches Beispiel: **/tmp**

Anzeige in ls -l:

- **t** statt **x** beim Anderen-Ausführbit: **drwxrwxrwt**
- **T** wenn Sticky gesetzt aber kein Ausführrecht für Andere: **drwxrwxrwT**

```
# Sticky Bit auf Verzeichnis setzen
chmod +t /tmp/geteilt
chmod 1777 /tmp/geteilt

# Sticky Bit entfernen
chmod -t /tmp/geteilt

# Prüfen
ls -ld /tmp

# Alle Verzeichnisse mit Sticky Bit finden
find / -type d -perm -1000 -ls 2>/dev/nulldrwxrwxrwt 10 root root
4096 Apr 29 10:00 /tmp
         ^
         t = Sticky Bit + Ausführrecht für Andere
```

Übersicht: Spezialbits

Bit	Name	Oktal	Symbolisch	Auf Dateien	Auf Verzeichnisse
SUID	Set-UID	4000	u+s	Läuft als Eigentümer der Datei	Keine standardisierte Wirkung
SGID	Set-GID	2000	g+s	Läuft als Gruppe der Datei	Neue Dateien erben Gruppe des Verzeichnisses
Sticky	Sticky Bit	1000	+t	Veraltet, keine Wirkung	Nur Eigentümer/root kann löschen

Kombination mit normalen Rechten

```
# SUID + 755 = 4755
chmod 4755 /usr/local/bin/myprog
# -rwsr-xr-x

# SGID + 2775 = setgid-Verzeichnis für Team
chmod 2775 /srv/team/
# drwxrwsr-x

# Sticky + 1777 = /tmp-ähnliches Verzeichnis
chmod 1777 /tmp
# drwxrwxrwt
```

Sicherheitshinweis: *SUID/SGID auf Dateien sind häufige Angriffsvektoren (Privilege Escalation). Regelmäßige Überprüfung mit* ***find / -perm -4000*** *bzw.* ***-perm -2000*** *ist empfehlenswert.*

WC

Beschreibung: Zählt Zeilen, Wörter, Zeichen und Bytes in Dateien oder der Standardeingabe. Wird sehr oft mit Pipes kombiniert um schnelle Statistiken zu erhalten.

Syntax: wc [OPTIONEN] [DATEI...]

Flags

Flag	Langform	Bedeutung
-l	--lines	Nur Zeilenanzahl ausgeben
-w	--words	Nur Wortanzahl ausgeben
-c	--bytes	Byte-Anzahl ausgeben
-m	--chars	Zeichenanzahl ausgeben (Multibyte/Unicode-sicher, unterscheidet sich von -c bei UTF-8)
-L	--max-line-length	Länge der längsten Zeile ausgeben

Reihenfolge der Standardausgabe

Ohne Flags gibt **wc** immer **drei Werte** aus in dieser Reihenfolge:

ZEILEN WÖRTER BYTES DATEINAME

Beispiele

```
# Standard: Zeilen, Wörter, Bytes
wc datei.txt

# Nur Zeilen zählen
wc -l datei.txt

# Nur Wörter zählen
wc -w datei.txt

# Nur Bytes zählen
wc -c datei.txt

# Zeichenanzahl (wichtig bei Umlauten/Unicode)
wc -m datei.txt

# Längste Zeile finden
wc -L datei.txt

# Mehrere Dateien + Gesamtsumme
wc -l *.txt

# Anzahl der Prozesse zählen
ps aux | wc -l

# Anzahl der Benutzer im System
wc -l < /etc/passwd

# Anzahl der Dateien in einem Verzeichnis
ls | wc -l

# Anzahl der Treffer einer Suche
grep -r "TODO" . | wc -l

# Datei auf maximale Zeilenlänge prüfen
wc -L skript.sh
```

Ausgabe

```
# wc datei.txt
  42  187  1204 datei.txt
#  |    |     |
#  |    |     +--  Bytes
#  |    +--------  Wörter
#  +-------------  Zeilen

# wc -l datei.txt
42 datei.txt

# wc -l *.txt
  10 datei1.txt
  32 datei2.txt
  42 gesamt

# ps aux | wc -l
127

# ls | wc -l
23

# wc -c vs wc -m (UTF-8: Umlaute)
echo "Straße" | wc -c
8                <- 8 Bytes (ß = 2 Bytes in UTF-8)
echo "Straße" | wc -m
7                <- 7 Zeichen
```

head

Beschreibung: Gibt die **ersten Zeilen** einer Datei aus. Standardmäßig werden die ersten 10 Zeilen angezeigt. Nützlich zum schnellen Einblick ohne alles laden zu müssen.

Syntax: head [OPTIONEN] [DATEI...]

Flags

Flag	Bedeutung
-n N oder -N	Erste N Zeilen ausgeben
-n -N	Alle Zeilen außer die letzten N ausgeben
-c N	Erste N Bytes ausgeben
-c -N	Alle Bytes außer die letzten N
-q	Dateinamen-Header unterdrücken (quiet, bei mehreren Dateien)
-v	Dateinamen-Header immer anzeigen (verbose, auch bei einzelner Datei)

Beispiele

```
# Erste 10 Zeilen (Standard)
head datei.txt

# Erste 5 Zeilen
head -n 5 datei.txt

# Kurzform: gleichbedeutend
head -5 datei.txt

# Alle Zeilen AUSSER die letzten 3
head -n -3 datei.txt

# Erste 100 Bytes der Datei
head -c 100 binärdatei

# Erstes Kilobyte einer Datei
head -c 1K datei.bin

# Erste Zeile einer Datei (z.B. CSV-Header lesen)
head -1 tabelle.csv

# Mehrere Dateien – mit Trennlinien dazwischen
head -n 3 datei1.txt datei2.txt

# Mehrere Dateien ohne Dateinamen-Header
head -q -n 3 *.conf

# In welchen Shell-Skripten steht bash als Shebang?
head -1 *.sh | grep "#!/bin/bash"
```

Ausgabe

```
# head -n 3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin

# head -n 3 datei1.txt datei2.txt
==> datei1.txt <==
Zeile 1 aus Datei 1
Zeile 2 aus Datei 1
Zeile 3 aus Datei 1

==> datei2.txt <==
Zeile 1 aus Datei 2
Zeile 2 aus Datei 2
Zeile 3 aus Datei 2

# head -n -3 datei.txt  (datei hat 6 Zeilen -> gibt Zeilen 1-3 aus)
Zeile 1
Zeile 2
Zeile 3
```

tail

Beschreibung: Gibt die **letzten Zeilen** einer Datei aus (Standard: 10). Besonders wertvoll zur **Echtzeit-Überwachung** von Log-Dateien mit **-f**.

Syntax: tail [OPTIONEN] [DATEI...]

Flags

Flag	Bedeutung
-n N oder -N	Letzte N Zeilen ausgeben
-n +N	Ab Zeile N bis zum Ende ausgeben (Zeilen 1 bis N-1 überspringen)
-c N	Letzte N Bytes ausgeben
-c +N	Ab Byte-Position N bis zum Ende
-f	Follow – Datei live verfolgen, neue Zeilen werden sofort angezeigt
-F	Wie -f, aber Datei wird neu geöffnet wenn sie rotiert/neu erstellt wird (robuster für Log-Rotation)
-s N	Intervall (Sekunden) zwischen Prüfungen bei -f (Standard: 1.0)
--pid=PID	Beendet sich automatisch wenn Prozess mit PID endet
-q	Dateinamen-Header unterdrücken
-v	Dateinamen-Header immer anzeigen

Beispiele

```
# Letzte 10 Zeilen (Standard)
tail datei.txt

# Letzte 20 Zeilen
tail -n 20 /var/log/syslog

# Kurzform
tail -20 /var/log/syslog

# Log-Datei live verfolgen (Strg+C zum Beenden)
tail -f /var/log/nginx/access.log

# Robusterer Follow (überlebt Log-Rotation)
tail -F /var/log/nginx/error.log

# Follow + gleichzeitig nach Fehlern filtern
tail -f /var/log/syslog | grep -i "error\|warn\|crit"

# Ab Zeile 50 bis Ende ausgeben
tail -n +50 datei.txt

# Letzten 1KB einer Datei ausgeben
tail -c 1K datei.txt

# Follow und automatisch beenden wenn Prozess endet
tail -f --pid=4321 /var/log/app.log

# Mehrere Log-Dateien gleichzeitig verfolgen
tail -f /var/log/syslog /var/log/auth.log

# Letzten Eintrag in CSV-Datei
tail -1 daten.csv
# Alle außer die erste Zeile ausgeben (Header überspringen)
tail -n +2 tabelle.csv
```

Ausgabe

```
# tail -n 3 /etc/passwd
matta:x:1000:1000:matta,,,:/home/matta:/bin/bash
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin

# tail -f /var/log/syslog  (live, neue Einträge erscheinen
automatisch)
Apr 29 10:00:01 server cron[1234]: (root) CMD (/usr/bin/backup.sh)
Apr 29 10:01:01 server cron[1235]: (root) CMD (/usr/bin/cleanup.sh)
Apr 29 10:02:00 server sshd[9876]: Accepted publickey for alice from
192.168.1.5
...   <- Strg+C zum Beenden

# tail -n +2 tabelle.csv  (Header wird übersprungen)
alice,30,Entwicklerin
bob,25,DevOps
charlie,35,Sysadmin
```

***Tipp: tail -f** ist das wichtigste Werkzeug für Live-Log-Analyse. **-F** ist stabiler bei Systemen mit Log-Rotation (z. B. **logrotate**).*

grep

Beschreibung: Sucht in Dateien oder der Standardeingabe nach Zeilen, die einem Muster (regulärer Ausdruck) entsprechen. "grep" steht für *Global Regular Expression Print*.

Syntax: grep [OPTIONEN] MUSTER [DATEI...]

Flags

Flag	Langform	Bedeutung
-i	--ignore-case	Groß-/Kleinschreibung ignorieren
-v	--invert-match	Invertieren – nur nicht-passende Zeilen ausgeben
-r	--recursive	Rekursiv in Verzeichnisbaum suchen
-R	--dereference-recursive	Wie -r, folgt aber auch symbolischen Links
-n	--line-number	Zeilennummer vor jede Ausgabezeile
-l	--files-with-matches	Nur Dateinamen ausgeben (nicht Inhalt)
-L	--files-without-match	Dateinamen ausgeben die NICHT passen
-c	--count	Anzahl der Treffer pro Datei ausgeben
-E	--extended-regexp	Erweiterte reguläre Ausdrücke (ERE)
-F	--fixed-strings	Feste Zeichenkette, kein Regex-Parsing
-P	--perl-regexp	Perl-kompatible Regex (PCRE)
-o	--only-matching	Nur den genau passenden Teil ausgeben
-A N	--after-context=N	N Zeilen nach dem Treffer ausgeben
-B N	--before-context=N	N Zeilen vor dem Treffer ausgeben
-C N	--context=N	N Zeilen vor und nach dem Treffer
-w	--word-regexp	Nur ganze Wörter suchen
-x	--line-regexp	Nur ganze Zeilen suchen
--color	--colour	Treffer farbig hervorheben
-m N	--max-count=N	Nach N Treffern abbrechen
-q	--quiet	Keine Ausgabe, nur Exit-Code (0 = gefunden)
-s	--no-messages	Fehlermeldungen unterdrücken
-h	--no-filename	Dateiname in Ausgabe unterdrücken
-H	--with-filename	Dateiname immer ausgeben

Reguläre Ausdrücke (Kurzübersicht)

Muster	Bedeutung
.	Ein beliebiges Zeichen
^	Zeilenanfang
$	Zeilenende
*	0 oder mehr des vorigen Zeichens
+	1 oder mehr (ERE: -E)
?	0 oder 1 (ERE: -E)
[abc]	Eines der Zeichen a, b oder c
[^abc]	Keines der Zeichen a, b, c
[a-z]	Zeichenklasse a bis z
\b	Wortgrenze
(a|b)	a oder b (ERE: -E)
{n,m}	n bis m Wiederholungen (ERE: -E)

Beispiele

```
# Einfache Suche
grep "root" /etc/passwd

# Groß-/Kleinschreibung ignorieren
grep -i "error" /var/log/syslog

# Rekursiv in einem Verzeichnis suchen
grep -r "TODO" /home/matta/projekt/

# Mit Zeilennummern
grep -n "bash" /etc/passwd

# Invertierte Suche: alle Nicht-Kommentar-Zeilen in Konfigdatei
grep -v "^#" /etc/ssh/sshd_config | grep -v "^$"

# Nur Dateinamen ausgeben (welche Dateien enthalten den Begriff?)
grep -rl "passwort" /etc/

# Anzahl der Treffer pro Datei
grep -rc "ERROR" /var/log/

# Erweiterte Regex: root oder daemon am Zeilenanfang
grep -E "^(root|daemon)" /etc/passwd

# Zeilen mit Kontext ausgeben
grep -C 2 "CRITICAL" /var/log/syslog

# Nur passende Teile ausgeben – alle IP-Adressen extrahieren
grep -oE "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}" access.log

# Nur ganze Wörter finden
grep -w "pass" datei.txt

# Stilles grep – nur Exit-Code prüfen (für Skripte)
if grep -q "error" /var/log/app.log; then
    echo "Fehler gefunden!"
fi

# Nach einem festen String suchen (kein Regex)
grep -F "192.168.1.1" /var/log/auth.log
```

```
# Maximal 5 Treffer ausgeben
grep -m 5 "warn" /var/log/syslog

# Mehrere Dateien, mit Dateinamen ausgeben
grep -H "PermitRootLogin" /etc/ssh/sshd_config
```

Ausgabe

```
# grep "root" /etc/passwd
root:x:0:0:root:/root:/bin/bash

# grep -n "bash" /etc/passwd
1:root:x:0:0:root:/root:/bin/bash
42:matta:x:1000:1000::/home/matta:/bin/bash

# grep -c "ERROR" /var/log/app.log
15

# grep -C 2 "CRITICAL" /var/log/syslog
Apr 29 09:58:01 server app[123]: Connection established
Apr 29 09:59:00 server app[123]: Retry attempt 3
Apr 29 09:59:30 server app[123]: CRITICAL: Database unreachable
Apr 29 09:59:31 server app[123]: Attempting failover...
Apr 29 09:59:45 server app[123]: Failover successful

# grep -oE "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
access.log
192.168.1.10
10.0.0.1
172.16.5.8
```

find

Beschreibung: Sucht rekursiv nach Dateien und Verzeichnissen anhand von Name, Typ, Größe, Datum, Berechtigungen, Eigentümer u. v. m. Kann gefundene Dateien direkt weiterverarbeiten.

Syntax: find [STARTPFAD…] [AUSDRUCK]

Wichtige Optionen

Typ und Name

Option	Bedeutung
-name MUSTER	Dateiname (case-sensitive, Wildcards z.B. *.txt)
-iname MUSTER	Wie -name, aber case-insensitive
-type f	Reguläre Dateien
-type d	Verzeichnisse
-type l	Symbolische Links
-type p	Named Pipes
-type s	Sockets
-type b	Block-Gerät
-type c	Zeichen-Gerät

Größe und Zeit

Option	Bedeutung
-size +N[c/k/M/G]	Größer als N (c=Bytes, k=1024B, M=Mega, G=Giga)
-size -N	Kleiner als N
-size N	Exakt N
-empty	Leere Dateien oder leere Verzeichnisse
-mtime N	Inhalt geändert vor genau N*24h
-mtime -N	Geändert innerhalb der letzten N Tage
-mtime +N	Geändert vor mehr als N Tagen
-atime N	Letzter Zugriff vor N Tagen
-ctime N	Status (Inode) geändert vor N Tagen
-newer DATEI	Neuer als angegebene Datei
-newermt DATUM	Neuer als angegebenes Datum

Eigentümer und Berechtigungen

Option	Bedeutung
-user NAME	Gehört diesem Benutzer
-group NAME	Gehört dieser Gruppe
-uid N	Numerische UID
-gid N	Numerische GID
-nouser	Kein gültiger Benutzer (verwaiste Dateien)
-nogroup	Keine gültige Gruppe
-perm MODUS	Exakt diese Berechtigungen
-perm -MODUS	Mindestens diese Bits gesetzt
-perm /MODUS	Mindestens eines dieser Bits gesetzt

Navigation

Option	Bedeutung
-maxdepth N	Maximal N Verzeichnisebenen tief suchen
-mindepth N	Erst ab Tiefe N suchen
-depth	Verzeichnisinhalt vor dem Verzeichnis selbst verarbeiten
-follow	Symbolischen Links folgen

Aktionen

Option	Bedeutung
-print	Pfad ausgeben (Standard)
-print0	Pfad ausgeben, mit Null-Byte getrennt (für xargs -0)
-ls	Ausführliches ls -l-Format ausgeben
-delete	Gefundene Dateien/leere Verzeichnisse löschen
-exec BEFEHL {} \;	Befehl für jede Datei einzeln ausführen
-exec BEFEHL {} +	Befehl einmal mit allen Dateien ausführen (effizienter)
-ok BEFEHL {} \;	Wie -exec, aber mit Bestätigungsabfrage

Logik

Option	Bedeutung
-not / !	Negation
-and / -a	UND (Standard, implizit)
-or / -o	ODER
\(AUSDRUCK \)	Gruppierung

Beispiele

```
# Alle .txt-Dateien ab aktuellem Verzeichnis
find . -name "*.txt"

# Alle .conf-Dateien in /etc (case-insensitive)
find /etc -iname "*.conf" -type f

# Leere Dateien finden
find /tmp -type f -empty

# Dateien größer als 100MB
find / -type f -size +100M -ls

# Dateien in den letzten 24 Stunden geändert
find /home -type f -mtime -1

# Dateien älter als 30 Tage im /tmp-Verzeichnis löschen
find /tmp -type f -mtime +30 -delete

# SUID-Bit-Dateien finden (Sicherheitscheck)
find / -type f -perm -4000 -ls 2>/dev/null

# SGID-Bit-Verzeichnisse finden
find / -type d -perm -2000 2>/dev/null

# Dateien mit genau Rechten 777 finden
find /tmp -perm 777 -type f

# Verwaiste Dateien finden (ohne gültigen Eigentümer)
find / -nouser -o -nogroup 2>/dev/null

# Dateien des Benutzers alice
find /home -user alice -type f

# Auf eine Verzeichnistiefe begrenzen
find /etc -maxdepth 1 -name "*.conf"

# Ausführen: alle .sh-Dateien ausführbar machen
find . -name "*.sh" -exec chmod +x {} \;

# Effizienter mit + (ein Aufruf für alle Dateien)
find . -name "*.log" -exec gzip {} +

# Mit xargs kombinieren (null-sicher)
find . -name "*.bak" -print0 | xargs -0 rm -f

# Nicht in .git-Verzeichnissen suchen
find . -not -path "./.git/*" -name "*.py"

# Dateien zwischen zwei Größen
find . -type f -size +1k -size -10k
# Mehrere Namen mit ODER
find . \( -name "*.jpg" -o -name "*.png" -o -name "*.gif" \) -type f
# Dateien neuer als eine Referenzdatei
find . -newer referenz.txt -type f
```

Ausgabe

```
# find . -name "*.txt" -type f
./dokumente/notizen.txt
./backup/alt.txt
./README.txt

# find /var/log -type f -mtime -1 -ls
12345678   8 -rw-r--r-- 1 syslog adm    7194 Apr 29 09:58
/var/log/syslog
12345679   4 -rw-r--r-- 1 root   adm    1302 Apr 29 08:00
/var/log/auth.log

# find / -type f -perm -4000 2>/dev/null
/usr/bin/passwd
/usr/bin/su
/usr/bin/sudo
/usr/bin/mount
```

awk

Beschreibung: Mächtiges Werkzeug zur strukturierten Textverarbeitung. **awk** liest Eingabe zeilenweise, teilt jede Zeile in Felder auf und wendet Muster-Aktions-Regeln an. Ideal für Logs, CSVs und **/etc/passwd**-ähnliche Dateien.

Syntax: awk [OPTIONEN] 'PROGRAMM' [DATEI...]

Grundstruktur eines awk-Programms

```
awk 'BEGIN { Initialisierungen }
   /MUSTER/ { Aktionen pro Zeile }
   END { Abschlussaktionen }' datei.txt
```

- **BEGIN** – wird einmal **vor** dem ersten Zeileneinlesen ausgeführt
- **END** – wird einmal **nach** dem letzten Zeileneinlesen ausgeführt
- Ohne Muster gilt die Aktion für **jede** Zeile

Eingebaute Variablen

Variable	Bedeutung
$0	Komplette aktuelle Zeile
$1, $2, ... $NF	Felder der Zeile (1-basiert)
NF	Anzahl der Felder in der aktuellen Zeile
NR	Aktuelle Zeilennummer (gesamt)
FNR	Zeilennummer innerhalb der aktuellen Datei
FS	Eingabe-Feldtrenner (Standard: Leerzeichen/Tab)
OFS	Ausgabe-Feldtrenner
RS	Eingabe-Zeilentrenner (Standard: \n)
ORS	Ausgabe-Zeilentrenner
FILENAME	Name der aktuellen Eingabedatei

Flags

Flag	Bedeutung
-F TRENNER	Feldtrenner festlegen (z.B. -F: oder -F",")
-v VAR=WERT	Variable vor Programmstart setzen
-f DATEI	awk-Programm aus Datei laden

Vergleichsoperatoren

Operator	Bedeutung
==	Gleich
!=	Ungleich
<, >	Kleiner/Größer
<=, >=	Kleiner-gleich/Größer-gleich
~	Regulärer Ausdruck passt
!~	Regulärer Ausdruck passt nicht

String-Funktionen

Funktion	Bedeutung
`length(s)`	Länge des Strings (ohne Arg: Länge von `$0`)
`substr(s, start [, len])`	Teilstring ab Position `start`, optional Länge `len`
`index(s, such)`	Position von `such` in `s` (0 = nicht gefunden)
`split(s, arr, trenn)`	`s` per Trenner in Array zerlegen; gibt Elementanzahl zurück
`sub(regex, ersatz [, s])`	Erste Übereinstimmung ersetzen (Standard: `$0`)
`gsub(regex, ersatz [, s])`	Alle Übereinstimmungen ersetzen; gibt Anzahl zurück
`match(s, regex)`	Regex in `s` suchen – setzt `RSTART` (Position) und `RLENGTH` (Länge)
`sprintf(format, ...)`	Formatierter String (gibt String zurück, druckt nicht)
`tolower(s)`	Alle Zeichen in Kleinbuchstaben umwandeln
`toupper(s)`	Alle Zeichen in Großbuchstaben umwandeln
`gensub(regex, ersatz, flag [, s])`	Erweitertes Ersetzen mit Rückreferenzen `\1` (gawk)

Hinweis: In `sub`/`gsub` steht `&` im Ersatzstring für den gesamten gefundenen Text.
Beispiel: `gsub(/[0-9]+/, "[&]")` → umschließt Zahlen mit eckigen Klammern.

Math-Funktionen

Funktion	Bedeutung
`int(x)`	Integer-Anteil (abschneiden, nicht runden)
`sqrt(x)`	Quadratwurzel
`sin(x)`, `cos(x)`	Sinus / Kosinus (Argument in Radiant)
`atan2(y, x)`	Arcustangens von y/x
`exp(x)`	e hoch x
`log(x)`	Natürlicher Logarithmus (Basis e)
`rand()`	Zufallszahl im Bereich [0, 1)
`srand([seed])`	Zufallszahlen-Seed setzen; gibt alten Seed zurück

Kontrollstrukturen

```
# if / else
    { if ($3 > 100) print "groß"; else print "klein" }

    # if / else if / else
    {
      if ($3 > 100) print "groß";
      else if ($3 > 50) print "mittel";
      else print "klein"
    }

    # for-Schleife (klassisch, über alle Felder einer Zeile
iterieren)
    { for (i = 1; i <= NF; i++) print i": "$i }

    # for-Schleife über Array-Schlüssel (Reihenfolge nicht
garantiert)
    END { for (key in count) print key, count[key] }

    # while-Schleife
    { i = 1; while (i <= NF) { print $i; i++ } }

    # do-while
    BEGIN { i = 1; do { print i; i++ } while (i <= 5) }

    # next: restliche Regeln überspringen, sofort nächste Zeile
holen
    /^#/ { next }    # Kommentarzeilen ignorieren
    { print }        # (wird für Kommentarzeilen nie erreicht)

    # exit: awk sofort beenden (END-Block wird trotzdem noch
ausgeführt)
    NR > 100 { exit }    # Nur die ersten 100 Zeilen verarbeiten
```

Beispiele

```
# Erste Spalte ausgeben
awk '{print $1}' datei.txt

# Letzte Spalte ausgeben (NF = Anzahl Felder)
awk '{print $NF}' datei.txt

# Benutzernamen aus /etc/passwd
awk -F: '{print $1}' /etc/passwd

# Benutzername + Home-Verzeichnis + Shell
awk -F: '{print $1, $6, $7}' /etc/passwd

# Eigener Ausgabe-Trennzeichen
awk -F: 'OFS=" | " {print $1, $3, $6}' /etc/passwd

# Zeilennummer + Inhalt
awk '{print NR": "$0}' datei.txt

# Nur Zeilen die "root" enthalten
awk '/root/ {print}' /etc/passwd

# Zeilen ausgeben bei denen Feld 3 (UID) größer 1000 ist
awk -F: '$3 > 1000 {print $1, $3}' /etc/passwd

# Zeilen ausgeben wo Shell NICHT /nologin ist
awk -F: '$7 !~ /nologin/ {print $1, $7}' /etc/passwd

# Summe einer Spalte berechnen
awk '{sum += $1} END {print "Summe:", sum}' zahlen.txt

# Durchschnitt berechnen
awk '{sum += $1; count++} END {print "Durchschnitt:", sum/count}'
zahlen.txt

# BEGIN und END mit Feldtrenner setzen
awk 'BEGIN{print "=== Start ==="; FS=":"} {print $1} END{print "===
Ende ==="}' /etc/passwd

# Variable übergeben
awk -v grenze=1000 -F: '$3 > grenze {print $1}' /etc/passwd

# printf für formatierte Ausgabe
awk -F: '{printf "%-15s UID: %4d\n", $1, $3}' /etc/passwd
# Feld ersetzen und neu ausgeben
awk -F: 'OFS=":" {$7="/bin/sh"; print}' /etc/passwd

# Zeilen zählen die ein Muster haben
awk '/ERROR/{count++} END{print count " Fehler gefunden"}'
/var/log/app.log

# Einzigartiger Trick: Duplikate entfernen (ohne sort)
awk '!seen[$0]++' datei.txt

# CSV ab Zeile 2 verarbeiten (Header überspringen)
awk 'NR > 1 {print $1, $3}' tabelle.csv
```

Ausgabe

```
# awk -F: '{print $1}' /etc/passwd
root
daemon
bin
matta
```

```
# awk -F: '$3 > 1000 {print $1, $3}' /etc/passwd
matta 1001
alice 1002

# awk -F: '{printf "%-15s UID: %4d\n", $1, $3}' /etc/passwd
root            UID:    0
daemon          UID:    1
matta           UID: 1000

# awk '{sum += $1; count++} END {print "Summe:", sum, "Schnitt:",
sum/count}' zahlen.txt
Summe: 165 Schnitt: 16.5
```

sed

Beschreibung: `sed` (Stream Editor) bearbeitet Text zeilenweise. Es liest Dateien oder stdin, wendet Transformationsregeln an und gibt das Ergebnis nach stdout aus. Ideal für automatisierte Textersetzung in Skripten.

Syntax: `sed [OPTIONEN] 'SKRIPT' [DATEI...]`

Alle Kommandozeilen-Optionen

Option	Langform	Bedeutung
`-n`	`--quiet` / `--silent`	Automatische Ausgabe unterdrücken; Ausgabe nur mit `p`
`-e SKRIPT`	`--expression=SKRIPT`	Skript direkt auf der Kommandozeile angeben
`-f DATEI`	`--file=DATEI`	sed-Programm aus einer Datei lesen
`-i[SUFFIX]`	`--in-place[=SUFFIX]`	Datei direkt bearbeiten (Backup mit optionalem Suffix)
`-E` / `-r`	`--regexp-extended`	Erweiterte reguläre Ausdrücke (ERE) aktivieren
`-s`	`--separate`	Dateien separat behandeln (NR wird je Datei zurückgesetzt)
`-z`	`--null-data`	NUL (`\0`) als Zeilentrenner statt `\n` (für `find -print0`)
`--sandbox`		Sicherer Modus: `e`, `r`, `w` sind verboten
`--posix`		Strikt POSIX-konform arbeiten

sed-Befehle (innerhalb des Skripts)

Befehl	**Bedeutung**
`s/REGEX/ERSATZ/FLAGS`	Substituieren (Ersetzen)
`d`	Zeile löschen (delete)
`p`	Zeile ausgeben (print) – extra zur Normalausgabe
`q [CODE]`	Verarbeitung beenden (quit), optionaler Exit-Code
`Q [CODE]`	Sofort beenden ohne letzte Zeile auszugeben
`i\TEXT`	TEXT vor der Zeile einfügen (insert)
`a\TEXT`	TEXT nach der Zeile anhängen (append)
`c\TEXT`	Zeile durch TEXT ersetzen (change)
`y/ZEICHEN/ERSATZ/`	Zeichen 1:1 austauschen (transliterate)
`=`	Aktuelle Zeilennummer ausgeben
`n`	Nächste Zeile in Pattern Space laden (und ausgeben)
`N`	Nächste Zeile anhängen (Pattern Space wird mehrzeilig)
`D`	Erste Zeile des Pattern Space löschen, von vorne beginnen
`P`	Erste Zeile des Pattern Space ausgeben
`r DATEI`	Inhalt von DATEI nach der aktuellen Zeile einfügen
`R DATEI`	Eine Zeile aus DATEI einfügen (gawk-Erweiterung)
`w DATEI`	Matching-Zeilen in DATEI schreiben
`l`	Zeile lesbar ausgeben (unsichtbare Zeichen als Escape)
`e`	Pattern Space als Shell-Befehl ausführen (GNU sed, unsicher!)
`: LABEL`	Sprungmarke definieren
`b [LABEL]`	Zu LABEL springen (oder an Ende, Schleife möglich)
`t [LABEL]`	Zu LABEL springen wenn seit letztem `t` ein `s` erfolgreich war
`T [LABEL]`	Zu LABEL springen wenn seit letztem `T` kein `s` erfolgreich war

s-Befehl Flags

Flag	Bedeutung
`g`	Alle Vorkommen ersetzen (global)
`N`	Nur das N-te Vorkommen ersetzen (z.B. `s/a/b/2`)
`p`	Ersetzte Zeile ausgeben (sinnvoll mit `-n`)
`i` / `I`	Groß-/Kleinschreibung ignorieren
`e`	Ergebnis als Shell-Befehl ausführen (GNU sed)
`w DATEI`	Ersetzte Zeilen in DATEI schreiben

Adressierung (Zeilen auswählen)

Adresse	Bedeutung
`N`	Nur Zeile N
`N,M`	Zeilen N bis M
`N~SCHRITT`	Ab Zeile N jede SCHRITT-te Zeile (z.B. `1~2` = alle ungeraden)
`0~SCHRITT`	Jede SCHRITT-te Zeile (z.B. `0~2` = alle geraden)
`/REGEX/`	Alle Zeilen, die REGEX entsprechen
`/REGEX/,/REGEX2/`	Von erster REGEX-Zeile bis REGEX2-Zeile (Range)
`N,/REGEX/`	Ab Zeile N bis zur ersten REGEX-Zeile
`$`	Letzte Zeile
`!`	Negation (z.B. `3!d` = alle AUSSER Zeile 3 löschen)

Reguläre Ausdrücke (BRE vs ERE)

Ausdruck	**BRE (Standard)**	**ERE (`-E`)**
Gruppe	`\(…\)`	`(…)`
Oder	`\(a|b\)`	`(a|b)`
1 oder mehr	`a*` (mind. 0)	`a+`
0 oder 1	(kein direktes)	`a?`
N-mal	`a\{3\}`	`a{3}`
N bis M mal	`a\{2,5\}`	`a{2,5}`
Rückreferenz	`\1`, `\2`	`\1`, `\2`
Beliebig	`.`	`.`
Zeilenanfang	`^`	`^`
Zeilenende	`$`	`$`

Beispiele

```
# ── Ersetzen (s-Befehl) ──────────────────────────────────────
# Erstes Vorkommen pro Zeile ersetzen
sed 's/alt/neu/' datei.txt

# Alle Vorkommen ersetzen (g = global)
sed 's/alt/neu/g' datei.txt

# Nur das 2. Vorkommen pro Zeile ersetzen
sed 's/alt/neu/2' datei.txt

# Case-insensitive ersetzen
sed 's/fehler/FEHLER/gi' datei.txt

# In-place bearbeiten (Datei direkt ändern)
sed -i 's/alt/neu/g' datei.txt

# In-place mit Backup (Datei.bak wird angelegt)
sed -i.bak 's/alt/neu/g' datei.txt

# Mehrere Befehle mit -e
sed -e 's/foo/bar/g' -e 's/baz/qux/g' datei.txt

# Mehrere Befehle mit Semikolon
sed 's/foo/bar/g; s/baz/qux/g' datei.txt

# sed-Programm aus Datei
sed -f korrekturen.sed datei.txt

# ── Zeilen löschen (d-Befehl) ────────────────────────────────
# Zeilen mit "TODO" löschen
sed '/TODO/d' datei.txt

# Kommentarzeilen (# am Anfang) löschen
sed '/^#/d' datei.txt
sed '/^[[:space:]]*#/d' datei.txt # auch eingerückte Kommentare
```

```
# Leerzeilen löschen
sed '/^$/d' datei.txt

# Leerzeilen und Zeilen mit nur Leerzeichen löschen
sed '/^s*$/d' datei.txt

# Erste Zeile löschen (z.B. Header)
sed '1d' datei.txt

# Letzte Zeile löschen
sed '$d' datei.txt

# Zeilen 3 bis 7 löschen
sed '3,7d' datei.txt

# Alle anderen Zeilen außer Zeile 5 löschen
sed '5!d' datei.txt

# ── Ausgabe filtern (-n   p) ──────────────────────────────
# Nur Matching-Zeilen ausgeben (wie grep)
sed -n '/muster/p' datei.txt

# Zeilen 5 bis 10 ausgeben
sed -n '5,10p' datei.txt

# Erste Zeile ausgeben (wie head -1)
sed -n '1p' datei.txt

# Letzte Zeile ausgeben (wie tail -1)
sed -n '$p' datei.txt

# Zeilen ausgeben, die NICHT dem Muster entsprechen (wie grep -v)
sed -n '/muster/!p' datei.txt

# ── Einfügen und Anhängen (i, a, c) ──────────────────────
# Text vor Zeile 3 einfügen
sed '3iDies kommt vor Zeile 3' datei.txt

# Text nach Zeile 3 anfügen
sed '3aDies kommt nach Zeile 3' datei.txt

# Zeile 3 komplett ersetzen
sed '3cDas ist jetzt Zeile 3' datei.txt

# Text nach jeder Zeile mit "START" einfügen
sed '/START/a---------' datei.txt

# Dateiinhalt nach bestimmter Zeile einfügen
sed '/INSERT_HERE/r template.txt' datei.txt

# ── Whitespace bereinigen ─────────────────────────────────
# Leerzeichen am Zeilenende entfernen (trailing whitespace)
sed 's/[[:space:]]*$//' datei.txt

# Führende Leerzeichen entfernen (leading whitespace)
sed 's/^[[:space:]]*//' datei.txt

# Beides (trim)
sed 's/^[[:space:]]*//; s/[[:space:]]*$//' datei.txt

# Tabulatoren durch Leerzeichen ersetzen
sed 's/t/    /g' datei.txt
```

```
# ─── Zeilenoperationen ──────────────────────────────────────
# Zeilennummer vor jede Zeile (wie cat -n)
sed '=' datei.txt | sed 'N; s/n/t/'

# Zeilen ab 5 ausgeben (bis Ende)
sed -n '5,$p' datei.txt

# Script nach N Zeilen beenden (wie head -5)
sed '5q' datei.txt

# Jede 2. Zeile löschen (gerade Zeilennummern)
sed '0~2d' datei.txt

# Jede 2. Zeile löschen (ungerade Zeilennummern)
sed '1~2d' datei.txt

# ─── Rückreferenzen ─────────────────────────────────────────
# Wort in Anführungszeichen einschließen (BRE)
sed 's/(Hallo)/"1"/' datei.txt

# Dasselbe mit ERE (-E)
sed -E 's/(Hallo)/"1"/' datei.txt

# Datum von YYYY-MM-DD zu DD.MM.YYYY umformatieren
sed -E 's/([0-9]{4})-([0-9]{2})-([0-9]{2})/3.2.1/' datei.txt

# Vornamen und Nachnamen tauschen ("Muster, Hans" → "Hans Muster")
sed -E 's/([A-Za-z] ), ([A-Za-z] )/2 1/' namen.txt

# ─── Zeichenersatz (y-Befehl) ───────────────────────────────
# Kleinbuchstaben in Großbuchstaben
sed 'y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/'
datei.txt

# Umlaute vereinfachen
sed 'y/äöüÄÖÜ/aouAOU/' datei.txt

# ─── Mehrzeilige Operationen (N-Befehl) ─────────────────────
# Zeilenumbruch zwischen zusammengehörigen Zeilen entfernen
sed 'N; s/n/ /' datei.txt

# ─── Praktische Anwendungen ─────────────────────────────────
# HTML-Tags entfernen
sed 's/<[^>]*>//g' seite.html

# DOS-Zeilenenden (rn) in Unix-Zeilenenden (n) umwandeln
sed 's/r//' datei.txt

# Alle IP-Adressen anonymisieren
sed -E 's/[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}/X.X.X.X/g'
access.log

# Passwort in Konfigurationsdatei ersetzen (sicher mit Backup)
sed -i.bak "s/^password=.*/password=NEU/" /etc/app/config.ini

# Leerzeilen auf maximal eine reduzieren
sed '/^$/{ N; /^n$/d }' datei.txt
# oder einfacher (GNU sed):
sed -E '/^$/{ /^n $/d }' datei.txt
```

Ausgabe

```
# sed 's/root/ROOT/g' /etc/passwd | head -2
ROOT:x:0:0:ROOT:/ROOT:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin

# sed -n '1,3p' /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin

# echo "2026-05-06" | sed -E 's/([0-9]{4})-([0-9]{2})-([0-
9]{2})/3.2.1/'
06.05.2026

# sed '=' datei.txt | sed 'N; s/\n/\t/'
1 Zeile eins
2 Zeile zwei
3 Zeile drei

# sed -i.bak 's/DEBUG/INFO/g' app.log
# (app.log wird verändert, app.log.bak enthält das Original)
```

Tipp: Vor jedem `sed -i` (in-place) immer mit `sed -n ... p` oder ohne `-i` testen, damit keine Daten versehentlich zerstört werden!

ls

Beschreibung: Listet den Inhalt von Verzeichnissen auf. Eines der meistgenutzten Linux-Kommandos.

Syntax: ls [OPTIONEN] [PFAD...]

Flags

Flag	Langform	Bedeutung
-l	--format=long	Lange Listenansicht mit Rechten, Eigentümer, Größe, Datum
-a	--all	Alle Dateien anzeigen, auch versteckte (.-Dateien)
-A	--almost-all	Wie -a, aber ohne . und ..
-h	--human-readable	Größen menschenlesbar (KB, MB, GB) – nur mit -l
-R	--recursive	Rekursiv alle Unterverzeichnisse anzeigen
-t	–	Nach Änderungszeit sortieren (neueste zuerst)
-r	--reverse	Sortierung umkehren
-S	–	Nach Dateigröße sortieren (größte zuerst)
-X	–	Nach Dateiendung sortieren
-i	--inode	Inode-Nummer ausgeben
-d	--directory	Verzeichnis selbst anzeigen, nicht seinen Inhalt
-1	–	Eine Datei pro Zeile ausgeben
-m	–	Kommagetrennte Liste ausgeben
--color	--colour	Farbige Ausgabe (Dateitypen farbig)
-F	--classify	Typkennzeichen anhängen (/ Verz., * ausführbar, @ Link, `
-n	--numeric-uid-gid	Numerische UIDs/GIDs statt Namen
-G	--no-group	Gruppenname in Langansicht weglassen
-Z	–	SELinux-Kontext anzeigen

Erklärung der Langansicht (ls -l)

```
drwxr-xr-x  2  matta  matta  4096  Apr 29 10:00  dokumente
|          | |     |     |    |           |
|          | |     |     |    |           +-- Dateiname
|          | |     |     |    +-- Datum der letzten Änderung
|          | |     |     +-- Dateigröße (Bytes)
|          | |     +-- Gruppe
|          | +-- Eigentümer
|          +-- Anzahl der Hard Links
+-- Typ + Berechtigungen:
   d = Verzeichnis
   - = Reguläre Datei
   l = Symbolischer Link
   c = Zeichengerät
   b = Blockgerät
   p = Named Pipe
   s = Socket
```

Dateitypen durch Farben (bei --color=auto)

Farbe	Typ
Keine/Weiß	Reguläre Datei
Blau	Verzeichnis
Cyan	Symbolischer Link
Grün	Ausführbare Datei
Rot	Archivdatei
Magenta	Bild/Mediendatei
Gelb/Blinkend	Gerätedatei
Rot auf Schwarz	Defekter Symlink

Beispiele

```
# Einfache Liste
ls

# Lange Ansicht
ls -l

# Alle Dateien inkl. versteckte + lange Ansicht + lesbare Größen
ls -lah

# Alle Dateien, nach Zeit sortiert (neueste zuerst)
ls -laht

# Alle Dateien, nach Größe sortiert (größte zuerst)
ls -lahS

# Alle Dateien, nach Zeit sortiert (älteste zuerst)
ls -latr

# Rekursiv alle Inhalte anzeigen
ls -lR /etc/ | head -30

# Inode-Nummern anzeigen (nützlich für Hard Links)
ls -li

# Verzeichnis selbst anzeigen (nicht seinen Inhalt)
ls -ld /etc/

# Nur Verzeichnisse im aktuellen Verzeichnis
ls -d */

# Typkennzeichen anzeigen
ls -F

# Numerische UIDs/GIDs
ls -ln

# Nur Verzeichnisse (grep auf d am Anfang)
ls -la | grep "^d"

# Nur Symlinks
ls -la | grep "^l"

# Alle ausführbaren Dateien
ls -la | grep "^-.*x"
```

Ausgabe

```
# ls -lah
gesamt 48K
drwxr-xr-x  5 matta matta 4,0K Apr 29 10:00 .
drwxr-xr-x 18 matta matta 4,0K Apr 28 09:00 ..
-rw-r--r--  1 matta matta  220 Apr 27 08:00 .bash_logout
-rw-r--r--  1 matta matta 3,5K Apr 27 08:00 .bashrc
drwxr-xr-x  2 matta matta 4,0K Apr 29 09:30 dokumente
-rw-r--r--  1 matta matta 2,1K Apr 29 09:45 notizen.txt
-rwxr-xr-x  1 matta matta 1,2K Apr 29 10:00 skript.sh
lrwxrwxrwx  1 matta matta   11 Apr 29 10:00 link -> skript.sh

# ls -li   (gleiche Inode = Hard Links)
524292 drwxr-xr-x 2 matta matta 4096 Apr 29 09:30 dokumente
524293 -rw-r--r-- 2 matta matta 1234 Apr 29 09:45 datei.txt
524293 -rw-r--r-- 2 matta matta 1234 Apr 29 09:45 hardlink.txt
#      ^^^^^^^^^^  Gleiche Inode = Hard Links!
```

pwd

Beschreibung: Gibt den **absoluten Pfad** des aktuellen Arbeitsverzeichnisses aus. "pwd" steht für *Print Working Directory*.

Syntax: pwd [OPTIONEN]

Flags

Flag	Bedeutung
-L	Logischer Pfad (Standard): zeigt den Pfad wie er in der Shell geführt wird, Symlinks werden nicht aufgelöst
-P	Physischer Pfad: löst alle Symlinks auf und zeigt den echten Pfad im Dateisystem

Unterschied -L vs. -P

Wenn **/home/matta** ein symbolischer Link auf **/data/users/matta** ist:

```
cd /home/matta

pwd      # oder pwd -L
# /home/matta          <- logischer Pfad (Symlink selbst)

pwd -P
# /data/users/matta    <- physischer Pfad (echtes Ziel)
```

Beispiele

```
# Aktuelles Verzeichnis ausgeben
pwd

# Physischen Pfad (Symlinks aufgelöst)
pwd -P

# In Skripten: aktuelles Verzeichnis speichern und später
zurückkehren
orig=$(pwd)
cd /tmp
echo "Jetzt in: $(pwd)"
cd "$orig"
echo "Zurück in: $(pwd)"

# Prüfen ob man sich im richtigen Verzeichnis befindet
if [ "$(pwd)" != "/var/www/html" ]; then
    echo "Falsches Verzeichnis!"
    exit 1
fi

# In Kombination mit find (absoluter Pfad als Startpunkt)
find "$(pwd)" -name "*.conf"
```

Ausgabe

```
# pwd
/home/matta/projekte/webserver

# Wenn CWD ein Symlink ist:
# pwd -L
/home/matta           <- zeigt Symlink-Pfad

# pwd -P
/data/users/matta     <- zeigt echten Pfad
```

sort

Beschreibung: Sortiert Zeilen eines Textes alphabetisch, numerisch oder nach anderen Kriterien.

Syntax: sort [OPTIONEN] [DATEI...]

Flags

Flag	Langform	Bedeutung
-r	--reverse	Umgekehrte Reihenfolge
-n	--numeric-sort	Numerisch sortieren (nicht alphabetisch: 10 > 9)
-h	--human-numeric-sort	Menschenlesbare Zahlen sortieren (1K < 1M < 1G)
-f	--ignore-case	Groß-/Kleinschreibung ignorieren
-u	--unique	Duplikate entfernen
-b	--ignore-leading-blanks	Führende Leerzeichen ignorieren
-k N[,M]	--key=N[,M]	Sortierschlüssel: Ab Feld N (bis Feld M)
-t TRENNER	--field-separator	Spaltentrenner festlegen
-o DATEI	--output=DATEI	Ergebnis in Datei schreiben (kann Eingabedatei sein!)
-c	--check	Prüfen ob Eingabe bereits sortiert ist
-C	–	Wie -c, aber ohne Fehlermeldung
-m	--merge	Bereits sortierte Dateien zusammenführen
-R	--random-sort	Zufällige Reihenfolge
-V	--version-sort	Versionsnummern sortieren (1.9 < 1.10)
-z	--zero-terminated	Null-Byte als Zeilentrenner (für find -print0)
-S N	--buffer-size=N	Puffergröße (z.B. -S 1G)
--parallel=N	–	N CPU-Kerne nutzen

Sortierschlüssel -k verstehen

sort -k ANFANG[,ENDE][OPTIONEN]

Beispiele:

-k2 Ab Feld 2 bis Zeilenende sortieren
-k2,2 Nur Feld 2 als Schlüssel
-k2,2n Feld 2 numerisch sortieren
-k3,3r Feld 3 rückwärts sortieren
-k2,2 -k1,1 Erst nach Feld 2, dann nach Feld 1

Beispiele

```
# Alphabetisch sortieren
sort namen.txt

# Numerisch sortieren (wichtig: sonst 10 < 2 alphabetisch!)
sort -n zahlen.txt

# Umgekehrt alphabetisch
sort -r namen.txt

# Numerisch + umgekehrt (größte Zahl oben)
sort -nr zahlen.txt

# Groß-/Kleinschreibung ignorieren
sort -f namen.txt

# Duplikate entfernen
sort -u namen.txt

# /etc/passwd nach UID (3. Feld, numerisch) sortieren
sort -t: -k3,3n /etc/passwd

# Dateien nach Größe auflisten (du + sort)
du -sh /var/* | sort -h

# Größte Dateien/Verzeichnisse finden
du -sh /var/log/* | sort -rh | head -10

# Prozesse nach CPU-Verbrauch sortieren
ps aux | sort -k3 -rn | head -10

# IP-Adressen sortieren (nach Oktet)
sort -t. -k1,1n -k2,2n -k3,3n -k4,4n ip_liste.txt

# Versionsnummern sortieren
sort -V versionen.txt

# Prüfen ob Datei bereits sortiert ist
sort -c namen.txt && echo "Bereits sortiert" || echo "Nicht
sortiert"

# In Datei schreiben (kann auch Eingabedatei sein)
sort namen.txt -o namen.txt
```

Ausgabe

```
# cat namen.txt
Zara
Alice
bob
Alice

# sort namen.txt
Alice
Alice
Zara
bob         <- Großbuchstaben vor Kleinbuchstaben!

# sort -f namen.txt   (case-insensitive)
Alice
Alice
bob
Zara

# sort -u namen.txt
Alice
Zara
bob

# sort -t: -k3,3n /etc/passwd | head -4
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin

# du -sh /var/log/* | sort -rh | head -5
248M    /var/log/journal
12M     /var/log/syslog
4,5M    /var/log/auth.log
1,2M    /var/log/kern.log
440K    /var/log/dpkg.log
```

cat

Beschreibung: Gibt den Inhalt von Dateien aus, verbindet mehrere Dateien und leitet Inhalte weiter. "cat" steht für *concatenate* (verketten).

Syntax: cat [OPTIONEN] [DATEI...]

Flags

Flag	Langform	Bedeutung
-n	--number	Alle Zeilen nummerieren
-b	--number-nonblank	Nur nicht-leere Zeilen nummerieren (überschreibt -n)
-A	--show-all	Alles anzeigen: entspricht -vET (Tabs als ^I, Zeilenende als $)
-T	--show-tabs	Tabs als ^I sichtbar machen
-E	--show-ends	Zeilenenden als $ anzeigen
-v	--show-nonprinting	Nicht-druckbare Zeichen sichtbar machen
-s	--squeeze-blank	Mehrere aufeinanderfolgende Leerzeilen zu einer zusammenfassen
-e	–	Entspricht -vE
-t	–	Entspricht -vT

Umleitungsoperatoren

Operator	Bedeutung
>	Ausgabe in Datei umleiten (überschreibt!)
>>	Ausgabe an Datei anhängen
<	Eingabe aus Datei lesen
<<EOF	Here-Document: mehrzeilige Eingabe direkt im Skript

Beispiele

```
# Datei ausgeben
cat datei.txt

# Mit Zeilennummern
cat -n datei.txt

# Nur nicht-leere Zeilen nummerieren
cat -b datei.txt

# Tabs und Zeilenenden sichtbar machen (Debugging von Skripten)
cat -A skript.sh

# Leerzeilen zusammenfassen
cat -s datei.txt

# Mehrere Dateien verketten und ausgeben
cat datei1.txt datei2.txt datei3.txt

# Mehrere Dateien zu einer zusammenführen
cat datei1.txt datei2.txt > gesamt.txt

# Ans Ende einer Datei anhängen
cat neue_zeilen.txt >> bestehend.txt

# Neue Datei interaktiv erstellen (Strg+D = EOF)
cat > neue_datei.txt
Zeile 1
Zeile 2
^D

# Here-Document – mehrzeiligen Text direkt in Datei schreiben
cat > konfiguration.conf << EOF
# Automatisch generiert
host=localhost
port=8080
debug=false
EOF

# Leere Datei erstellen
cat /dev/null > datei.txt

# Binäre Dateien verketten (z.B. Split-Archive)
cat archiv.part1 archiv.part2 archiv.part3 > archiv.tar.gz

# Inhalt einer Datei in eine Variable einlesen
INHALT=$(cat konfiguration.conf)
```

Ausgabe

```
# cat -n datei.txt
     1  Zeile eins
     2  Zeile zwei
     3
     4  Zeile vier

# cat -b datei.txt   (leere Zeile bekommt keine Nummer)
     1  Zeile eins
     2  Zeile zwei

     3  Zeile vier

# cat -A skript.sh
#!/bin/bash$
echo "Hallo"$
^Ieingerückter Text$
# ^I = Tab-Zeichen, $ = Zeilenende

# cat -s datei.txt   (Leerzeilen zusammengefasst)
Zeile 1

Zeile 3
# (vorher waren 3 Leerzeilen zwischen Zeile 1 und 3)
```

tac

Beschreibung: Gibt den Inhalt einer Datei in **umgekehrter Zeilenreihenfolge** aus. Der Name ist "cat" rückwärts geschrieben.

Syntax: tac [OPTIONEN] [DATEI...]

Flags

Flag	Bedeutung
-s TRENNER	Eigenen Datensatz-Trennzeichen verwenden (statt Newline \n)
-r	Den Trenner als regulären Ausdruck interpretieren
-b	Trenner wird an den Anfang statt an das Ende gestellt

Beispiele

```
# Datei umgekehrt ausgeben
tac datei.txt

# Log-Datei umgekehrt anzeigen: neueste Einträge zuerst
tac /var/log/syslog | less

# Nur die letzten 20 Einträge (von hinten)
tac /var/log/auth.log | head -20

# Mit grep kombinieren: letzten Fehler suchen
tac /var/log/syslog | grep -m 1 "ERROR"

# History umgekehrt anzeigen (letzter Befehl zuerst)
history | tac | head -10

# Eigener Trenner (z.B. Datensätze durch "---" getrennt)
tac -s "---" mehrteilig.txt

# Zeilen umkehren und mit Zeilennummern versehen
tac datei.txt | cat -n
```

Ausgabe

```
# cat datei.txt
Zeile 1
Zeile 2
Zeile 3
Zeile 4

# tac datei.txt
Zeile 4
Zeile 3
Zeile 2
Zeile 1

# tac /var/log/auth.log | grep -m 1 "Failed"
Apr 29 09:45:01 server sshd[9876]: Failed password for invalid user
admin from 10.0.0.5
# <- letzter fehlgeschlagener Login-Versuch
```

passwd

Beschreibung: Verwaltet Benutzerpasswörter. Normale Benutzer können ihr **eigenes** Passwort ändern; root kann **alle** Passwörter und Konto-Ablaufzeiten verwalten. Das Passwort wird verschlüsselt in **/etc/shadow** gespeichert.

Syntax: passwd [OPTIONEN] [BENUTZER]

Flags

Flag	Langform	Bedeutung
-l	--lock	Konto sperren – fügt ! vor den Hash in /etc/shadow ein (Login mit Passwort nicht mehr möglich, SSH-Key-Login aber weiterhin!)
-u	--unlock	Konto entsperren – entfernt ! aus dem Hash
-d	--delete	Passwort löschen (leeres Passwort = unsicher!)
-e	--expire	Passwort sofort ablaufen lassen – Benutzer muss es beim nächsten Login ändern
-n TAGE	--minimum=TAGE	Mindestanzahl Tage bis Passwort geändert werden darf (0 = jederzeit)
-x TAGE	--maximum=TAGE	Maximale Gültigkeitsdauer in Tagen (99999 = unbegrenzt)
-w TAGE	--warning=TAGE	Warnung N Tage vor Ablauf
-i TAGE	--inactive=TAGE	Konto wird N Tage nach Ablauf deaktiviert
-S	--status	Passwortstatus anzeigen
--stdin	–	Passwort von Standardeingabe lesen (nicht auf allen Systemen verfügbar)

Status-Ausgabe (passwd -S)

```
BENUTZER  STATUS  DATUM       MIN  MAX  WARN  INAKTIV
alice     P       04/29/2026  0    99999 7    -1
```

Status-Codes:
P = Password (Passwort gesetzt)
L = Locked (Konto gesperrt)
NP = No Password (kein Passwort)

Zusammenhang mit chage

chage bietet erweiterte Kontrolle über Passwort-Ablauf:

chage -l alice # Ablaufinformationen anzeigen
chage -M 90 alice # Maximale Gültigkeit 90 Tage
chage -E 2026-12-31 alice # Konto-Ablaufdatum setzen
chage -d 0 alice # Passwort sofort ablaufen lassen

Beispiele

```
# Eigenes Passwort ändern (interaktiv)
passwd

# Passwort eines Benutzers ändern (als root)
passwd alice

# Konto sperren (root)
passwd -l alice

# Konto entsperren
passwd -u alice

# Passwort sofort ablaufen lassen
passwd -e alice

# Passwortstatus anzeigen
passwd -S alice

# Minimale Gültigkeitsdauer setzen (7 Tage)
passwd -n 7 alice

# Maximale Gültigkeitsdauer setzen (90 Tage)
passwd -x 90 alice

# 14 Tage vorher warnen
passwd -w 14 alice

# 7 Tage nach Ablauf Konto deaktivieren
passwd -i 7 alice

# Passwort nicht-interaktiv setzen (sicher via chpasswd)
echo "alice:neuespasswort" | chpasswd
```

Ausgabe

```
# passwd alice
Neues Passwort:
Neues Passwort wiederholen:
passwd: Passwort erfolgreich aktualisiert

# passwd -S alice
alice P 04/29/2026 7 90 14 7
#     | |          |  |  |  +-- 7 Tage Inaktivität bis Deaktivierung
#     | |          |  |  +-- 14 Tage Warnung vor Ablauf
#     | |          |  +-- 90 Tage max. Gültigkeit
#     | |          +-- 7 Tage Mindestgültigkeit
#     | +-- Datum der letzten Änderung
#     +-- P = Passwort gesetzt
# passwd -l alice
passwd: Passwort für alice gesperrt.
# sudo grep alice /etc/shadow | cut -d: -f1,2
alice:!$6$hash...    <- ! vor dem Hash = gesperrt
```

set

Beschreibung: Das Shell-Built-in **set** hat zwei Hauptfunktionen:

1. **Shell-Optionen setzen** mit **-** (aktivieren) oder **+** (deaktivieren)
2. **Positionsparameter** (**$1**, **$2**, ...) neu setzen

set ist ein **Built-in der Shell** (bash, sh) – kein externes Programm. Es gilt für den aktuellen Shell-Prozess und alle daraus gestarteten Unterprozesse.

Syntax:

set [OPTIONEN] [ARGUMENTE]
set -o OPTIONSNAME <- langer Optionsname aktivieren
set +o OPTIONSNAME <- Option deaktivieren

***Merke: - aktiviert** eine Option, **+ deaktiviert** sie – das ist umgekehrt vom Intuitiven!*

Wichtige Optionen (Kurzform / Langform)

Kurzform	Langform (-o)	Bedeutung
-e	errexit	Skript sofort beenden wenn ein Befehl fehlschlägt (Exit-Code != 0)
-u	nounset	Fehler bei nicht gesetzten Variablen statt leerem String
-x	xtrace	Jeden Befehl vor Ausführung ausgeben (Debugging)
-v	verbose	Jede gelesene Eingabezeile ausgeben (vor Substitution)
-n	noexec	Befehle lesen und auf Syntaxfehler prüfen, aber nicht ausführen
-f	noglob	Glob-Expansion deaktivieren (*, ?, [...] werden nicht expandiert)
-C	noclobber	Vorhandene Dateien nicht mit > überschreiben
-b	notify	Status beendeter Hintergrundprozesse sofort melden
-h	hashall	Befehlspfade cachen (Standard: aktiviert)
-m	monitor	Job-Control aktivieren (Standard in interaktiver Shell)
-t	onecmd	Nach einem Befehl beenden
–	pipefail	Pipe schlägt fehl wenn irgendein Befehl in der Pipe fehlschlägt (kein Kurzform!)
–	nocaseglob	Glob-Matching ohne Groß-/Kleinschreibung
–	nullglob	Nicht-passende Globs werden zu leerem String
–	extglob	Erweiterte Glob-Muster aktivieren (*(...), +(...), etc.)
–	errtrace	-e gilt auch innerhalb von Funktionen

Die "Sichere Skript-Präambel"

```
#!/bin/bash
set -euo pipefail
```

Dies sind die **drei wichtigsten Optionen** für robuste Shell-Skripte:

Option	Was sie verhindert
-e	Skript läuft weiter obwohl ein Befehl fehlgeschlagen ist
-u	Eine nicht gesetzte Variable wird stillschweigend als leer behandelt
pipefail	Fehler in der Mitte einer Pipe wird ignoriert

Positionsparameter setzen

```
set -- alice bob charlie
echo $1   # alice
echo $2   # bob
echo $3   # charlie
echo $@   # alice bob charlie
echo $#   # 3 (Anzahl)

# "--" löst das Problem wenn Argumente mit - beginnen
set -- -n -v datei.txt
echo $1   # -n (ohne -- würde das als Option interpretiert)
```

Aktuelle Einstellungen anzeigen

```
# Alle gesetzten Optionen anzeigen
set -o

# Alle Variablen und Funktionen ausgeben
set

# Aktive Optionen als Buchstabenstring prüfen
echo $-

# Prüfen ob eine bestimmte Option aktiv ist
[[ $- == *e* ]] && echo "errexit aktiv"
```

Beispiele

```
# Skript sofort bei Fehler beenden
set -e

# Fehler bei ungesetzten Variablen
set -u

# Pipe-Fehler aktivieren
set -o pipefail

# Debugging: jeden Befehl vor Ausführung anzeigen
set -x
ls /tmp
set +x   # Debugging wieder deaktivieren

# Syntaxcheck ohne Ausführung
set -n

# Alle drei auf einmal setzen (empfohlene Praxis)
set -euo pipefail

# Glob-Expansion deaktivieren (* als Literal behandeln)
set -f
echo *.txt    # gibt "*.txt" aus – keine Dateinamen
set +f
echo *.txt    # gibt Dateinamen aus

# Überschreiben mit > verhindern
set -C
echo "test" > bereits_vorhanden.txt   # Fehler: Datei existiert
set +C
echo "test" > bereits_vorhanden.txt   # Erfolgreich

# Positionsparameter neu setzen
set -- eins zwei drei
echo "Anzahl: $#"   # 3
echo "Alle: $@"     # eins zwei drei

# Fehler für einen einzelnen Befehl erlauben (|| true verhindert -e-Abbruch)
befehl_der_fehlschlagen_kann || true

# Alternative: Fehlerbehandlung mit if (ignoriert -e automatisch)
if ! befehl_der_fehlschlagen_kann; then
    echo "Fehler aufgetreten, aber Skript läuft weiter"
fi
```

Ausgabe

```
# set -x  (Debugging aktiv)
# Jeder Befehl wird mit + am Anfang gezeigt
ls /tmp
+ ls /tmp
datei1.txt  datei2.txt

# set -o  (alle Optionen anzeigen)
allexport       off
braceexpand     on
errexit         off
errtrace        off
functrace       off
hashall         on
histexpand      on
history         on
ignoreeof       off
interactive-comments  on
keyword         off
monitor         on
noclobber       off
noexec          off
noglob          off
nolog           off
notify          off
nounset         off
onecmd          off
physical        off
pipefail        off
posix           off
privileged      off
verbose         off
vi              off
xtrace          off

# echo $-
himBHs    <- aktive Optionen als Buchstaben
```

Praxisbeispiel: Robustes Backup-Skript

```
#!/bin/bash
set -euo pipefail    # Sicher und robust

QUELLE="/home/matta"
ZIEL="/backup/matta"

echo "Sicherung von $QUELLE nach $ZIEL"

# Ohne -e würde das Skript weiterlaufen auch wenn mkdir fehlschlägt
mkdir -p "$ZIEL"

# Ohne pipefail würde ein Fehler in tar ignoriert wenn gzip
funktioniert
tar czf - "$QUELLE" | gzip > "$ZIEL/backup_$(date +%F).tar.gz"

echo "Fertig!"
```

***Wichtig: set -e** hat Ausnahmen – Befehle in **if**-Bedingungen, **while**-Tests sowie hinter **||** oder **&&** lösen keinen Exit aus, da ihr Fehlschlag dort erwartet wird.*

rsync

Beschreibung: `rsync` (Remote Sync) synchronisiert Dateien und Verzeichnisse lokal oder über Netzwerk. Es überträgt dank **Delta-Transfer-Algorithmus** nur die tatsächlich geänderten Teile einer Datei – extrem schnell und bandbreitenschonend.

Syntax: `rsync [OPTIONEN] QUELLE ZIEL`

Wie rsync intern funktioniert

```
    1. Zieldatei → Aufteilung in gleich große Blöcke
    2. Für jeden Block: schwache Prüfsumme (rolling checksum) + starke
Prüfsumme (MD5)
    3. Prüfsummen → Sender
    4. Sender gleicht Quelldatei mit den Prüfsummen ab
    5. Nur NEUE / GEÄNDERTE Blöcke werden übertragen
    6. Zieldatei wird aus alten + neuen Blöcken zusammengesetzt

    Ergebnis: Statt 500 MB werden vielleicht nur 3 KB übertragen.
```

Alle wichtigen Optionen

Option	Langform	Bedeutung
`-a`	`--archive`	Archivmodus = `-rlptgoD` zusammengefasst (empfohlen)
`-r`	`--recursive`	Verzeichnisse rekursiv übertragen
`-l`	`--links`	Symlinks als Symlinks übertragen
`-p`	`--perms`	Berechtigungen erhalten
`-t`	`--times`	Zeitstempel erhalten
`-g`	`--group`	Gruppeninfos erhalten
`-o`	`--owner`	Eigentümer erhalten (nur root)
`-D`		Gerätedateien + Sonderdateien erhalten
`-v`	`--verbose`	Ausführliche Ausgabe
`-vv`		Sehr ausführliche Ausgabe
`-q`	`--quiet`	Keine Ausgabe außer Fehlern
`-n`	`--dry-run`	Simulation – zeigt was passieren würde, ändert nichts
`-z`	`--compress`	Daten vor Übertragung komprimieren
`-P`		`--partial --progress` kombiniert: Fortschritt + Wiederaufnahme

Option	Langform	Bedeutung
`--progress`		Fortschrittsbalken für jede Datei
`--partial`		Unvollständige Übertragungen behalten (Wiederaufnahme möglich)
`-e BEFEHL`	`--rsh=BEFEHL`	Remote-Shell angeben (Standard: `ssh`)
`--delete`		Dateien im Ziel löschen, die in der Quelle fehlen (echtes Spiegeln)
`--delete-before`		Löschen VOR der Übertragung
`--delete-after`		Löschen NACH der Übertragung
`--exclude=MUSTER`		Dateien/Verzeichnisse ausschließen
`--exclude-from=DATEI`		Ausschlussliste aus Datei lesen
`--include=MUSTER`		Ausschluss für bestimmtes Muster aufheben
`--filter=REGEL`		Flexible Filterregel (`+ einschließen`, `- ausschließen`)
`--backup`		Backup geänderter Zieldateien anlegen
`--backup-dir=DIR`		Backup-Verzeichnis for geänderte Dateien
`--suffix=SUFFIX`		Backup-Suffix (Standard: `~`)
`-u`	`--update`	Neuere Zieldateien nicht überschreiben
`-c`	`--checksum`	Vergleich per Prüfsumme statt Größe + Mtime
`-H`	`--hard-links`	Hardlinks als Hardlinks erhalten
`-A`	`--acls`	ACLs (Access Control Lists) übertragen
`-X`	`--xattrs`	Erweiterte Attribute übertragen
`--chmod=RECHTE`		Berechtigungen an Zieldateien setzen
`--chown=USER:GRP`		Eigentümer/Gruppe an Zieldateien setzen
`--max-size=GRÖSSE`		Dateien über dieser Größe überspringen (z.B. `100M`)
`--min-size=GRÖSSE`		Dateien unter dieser Größe überspringen
`--bwlimit=KBPS`		Bandbreitenlimit (KB/s)
`--timeout=SEK`		I/O-Timeout in Sekunden
`--stats`		Übertragungsstatistik am Ende anzeigen
`-h`	`--human-readable`	Größen lesbar ausgeben (KB, MB, GB)
`--log-file=DATEI`		Übertragungsprotokoll in Datei schreiben

Option	Langform	Bedeutung
`--password-file=DATEI`		rsync-Daemon-Passwort aus Datei lesen
`--port=PORT`		Alternativen Port angeben (rsync-Daemon)
`--list-only`		Nur auflisten, nicht übertragen

Der Slash-Trick (sehr wichtig!)

```
rsync -av /quelle /ziel  # ← Kopiert ORDNER "quelle" in "ziel" → /ziel/quelle/...
rsync -av /quelle/ /ziel # ← Kopiert INHALT von "quelle" → /ziel/...
#                  ^
#                  Abschließender Slash bedeutet: "der Inhalt dieses Ordners"
```

Faustregel: Mit `/` am Ende der Quelle verhält sich rsync wie `cp -r quelle/* ziel/`.

Beispiele

```
# ── Lokal ──────────────────────────────────────────────────
# Einfache lokale Synchronisation
rsync -av /home/user/daten/ /backup/daten/

# Trockenlauf: erst prüfen, dann ausführen
rsync -avn /home/user/daten/ /backup/daten/

# Exaktes Spiegeln (überschüssige Zieldateien werden gelöscht)
rsync -av --delete /home/user/ /backup/user/

# Mit Fortschrittsanzeige
rsync -avP /quelle/ /ziel/
# gleichwertig:
rsync -av --progress --partial /quelle/ /ziel/

# ── Remote (SSH) ──────────────────────────────────────────
# Lokal → Remote (push)
rsync -avz /home/user/projekt/ benutzer@server:/backup/projekt/

# Remote → Lokal (pull)
rsync -avz benutzer@server:/var/www/html/ /lokal/webseite/

# SSH auf Port 2222
rsync -avz -e "ssh -p 2222" /daten/ user@server:/backup/

# SSH mit spezifischem Key
rsync -avz -e "ssh -i ~/.ssh/deploy_key" /daten/
user@server:/backup/

# ── Filtern ───────────────────────────────────────────────
# Temporäre Dateien und Logs ausschließen
rsync -av --exclude="*.tmp" --exclude="*.log" /daten/ /backup/daten/

# Ausschlussdatei (eine Regel pro Zeile)
# .rsync-exclude Inhalt: *.tmp / logs/ / .git/
rsync -av --exclude-from=/home/user/.rsync-exclude /daten/ /backup/
```

```
# Nur Bilder synchronisieren
rsync -av --include="*.jpg" --include="*.png" --exclude="*" /fotos/
/backup/fotos/

# node_modules ausschließen (typisch für Web-Projekte)
rsync -av --exclude="node_modules/" /projekt/ server:/deploy/

# ─── Backup mit Versionierung ──────────────────────────
# Geänderte Dateien mit Datum in separatem Ordner sichern
rsync -av --backup --backup-dir=/backup/$(date %Y-%m-%d) /home/
/backup/aktuell/

# Bandbreite begrenzen (500 KB/s)
rsync -avz --bwlimit=500 /daten/ user@server:/backup/

# Statistiken anzeigen
rsync -av --stats /quelle/ /ziel/

# Nur nach Prüfsumme vergleichen (langsamer, aber genauer)
rsync -avc /quelle/ /ziel/
```

Ausgabe

```
# rsync -avP /home/user/ /backup/
sending incremental file list
./
dokumente/
dokumente/bericht.pdf
     1,234,567 100%  12.34MB/s    0:00:00 (xfr#1, to-chk=23/47)
fotos/foto1.jpg
       987,654 100%   9.87MB/s    0:00:00 (xfr#2, to-chk=22/47)
...

sent 2,345,678 bytes  received 1,234 bytes  1,234,567.00 bytes/sec
total size is 45,678,901  speedup is 19.48
```

Tipp: Bei regelmäßigen Backups ist `rsync -avz --delete --backup --backup-dir=/backup/$(date +%F)` eine einfache aber effektive Incremental-Backup-Strategie.

basename

Beschreibung: Entfernt den Verzeichnispfad aus einem Dateipfad und gibt nur den **Dateinamen** zurück. Optional kann auch die Dateiendung abgeschnitten werden. Gegenstück: `dirname` (gibt den Verzeichnispfad zurück).

Syntax:

```
basename PFAD [SUFFIX]
basename -a PFAD... -s SUFFIX
```

Alle Optionen

Option	Langform	Bedeutung
`-a`	`--multiple`	Mehrere Pfade in einem Aufruf verarbeiten
`-s SUFFIX`	`--suffix=SUFFIX`	Dieses Suffix vom Ergebnis entfernen
`--help`		Hilfe anzeigen
`--version`		Versionsnummer anzeigen

Beispiele

```
# ─── Grundlegende Verwendung ──────────────────────────────
# Nur Dateiname (ohne Pfad)
basename /home/user/dokumente/bericht.pdf
# → bericht.pdf

# Dateiname ohne Endung
basename /home/user/dokumente/bericht.pdf .pdf
# → bericht

# Nur mit Programmname (Pfad egal)
basename /usr/bin/python3
# → python3

basename /usr/lib/systemd/systemd-journald
# → systemd-journald

# ─── Suffix entfernen (-s) ────────────────────────────────
# Endung .txt entfernen
basename -s .txt /pfad/zur/datei.txt
# → datei

# Doppeltes Suffix .tar.gz
basename /backup/archiv.tar.gz .tar.gz
# → archiv

# ─── Mehrere Dateien auf einmal (-a) ──────────────────────
basename -a /etc/hosts /etc/passwd /etc/shadow
# → hosts
#    passwd
#    shadow

# Mehrere Dateien   Suffix entfernen
basename -a -s .conf /etc/ssh/sshd_config.conf /etc/nginx/nginx.conf
# → sshd_config
#    nginx
```

```
# ─── In Skripten ──────────────────────────────────────────
# Skriptname ohne Pfad (für Fehlermeldungen)
PROG=$(basename "$0")
echo "Verwendung: $PROG [OPTIONEN]" >&2

# Lock-Datei basierend auf Skriptname
PROG=$(basename "$0" .sh)
LOCKFILE="/tmp/${PROG}.lock"
if [ -f "$LOCKFILE" ]; then
    echo "$PROG läuft bereits!" >&2
    exit 1
fi
touch "$LOCKFILE"
trap "rm -f '$LOCKFILE'" EXIT

# Alle .sh-Dateien ohne Endung verarbeiten
for f in /skripte/*.sh; do
    name=$(basename "$f" .sh)
    echo "Verarbeite Skript: $name"
done

# Backup mit Datum und originalem Dateinamen
DATEI="/etc/nginx/nginx.conf"
BACKUP="/backup/$(basename "$DATEI").$(date %Y%m%d)"
cp "$DATEI" "$BACKUP"
echo "Backup: $BACKUP"

# ─── dirname als Gegenstück ──────────────────────────────
DATEI="/home/user/daten/report.csv"
DIR=$(dirname "$DATEI")        # → /home/user/daten
NAME=$(basename "$DATEI" .csv) # → report

echo "Verzeichnis: $DIR"
echo "Dateiname:   $NAME"
echo "Voller Pfad: $DIR/$NAME.csv"
```

Ausgabe

```
# basename /home/matta/projekt/skript.sh
skript.sh

# basename /home/matta/projekt/skript.sh .sh
skript

# basename -a /var/log/syslog /var/log/auth.log /var/log/kern.log
syslog
auth.log
kern.log

# basename -a -s .log /var/log/syslog /var/log/auth.log
syslog
auth

# dirname /home/matta/projekt/skript.sh
/home/matta/projekt
```

docker

Beschreibung: Docker ist eine Plattform zur **Containerisierung von Anwendungen**. Ein Container ist eine isolierte, portable Ausführungsumgebung mit allem was die Anwendung benötigt (Code, Bibliotheken, Konfiguration).

Syntax: `docker [OPTIONEN] BEFEHL`

Grundkonzepte

Begriff	**Bedeutung**
Image	Unveränderliche Vorlage/Bauplan für Container
Container	Laufende (oder gestoppte) Instanz eines Images
Dockerfile	Textdatei mit Anweisungen zum Bauen eines Images
Registry	Speicher für Images (Docker Hub, ghcr.io, eigene)
Layer	Jede Dockerfile-Anweisung erzeugt eine unveränderliche Schicht
Volume	Persistenter Datenspeicher außerhalb des Containers
Network	Virtuelles Netzwerk für Container-Kommunikation

Wichtige docker-Befehle

Images

Befehl	**Bedeutung**
`docker pull IMAGE[:TAG]`	Image von Registry herunterladen
`docker push IMAGE[:TAG]`	Image in Registry hochladen
`docker build -t NAME .`	Image aus Dockerfile im aktuellen Verzeichnis bauen
`docker images`	Alle lokalen Images auflisten
`docker rmi IMAGE`	Image löschen
`docker tag QUELLE ZIEL`	Image neu taggen
`docker image prune`	Unbenutzte Images löschen
`docker image inspect IMAGE`	Detaillierte Image-Infos
`docker image history IMAGE`	Layer-Historie anzeigen
`docker save -o datei.tar IMAGE`	Image als Tar-Archiv exportieren
`docker load -i datei.tar`	Image aus Tar-Archiv laden

Container

Befehl	Bedeutung
`docker run IMAGE`	Container aus Image starten (erstellt neuen Container)
`docker start CONTAINER`	Gestoppten Container starten
`docker stop CONTAINER`	Container graceful stoppen (SIGTERM → SIGKILL)
`docker restart CONTAINER`	Container neu starten
`docker kill CONTAINER`	Container sofort beenden (SIGKILL)
`docker rm CONTAINER`	Gestoppten Container entfernen
`docker ps`	Laufende Container anzeigen
`docker ps -a`	Alle Container (auch gestoppte)
`docker exec CONTAINER BEFEHL`	Befehl in laufendem Container ausführen
`docker logs CONTAINER`	Container-Logs anzeigen
`docker inspect CONTAINER`	Detaillierte Container-Infos (JSON)
`docker stats`	Ressourcenverbrauch live anzeigen
`docker top CONTAINER`	Prozesse im Container anzeigen
`docker diff CONTAINER`	Dateisystemänderungen seit Start
`docker cp CONTAINER:PFAD LOKAL`	Datei aus Container kopieren
`docker commit CONTAINER IMAGE`	Container in Image umwandeln

Volumes und Netzwerke

Befehl	Bedeutung
`docker volume create NAME`	Volume erstellen
`docker volume ls`	Volumes auflisten
`docker volume rm NAME`	Volume löschen
`docker volume inspect NAME`	Volume-Details
`docker volume prune`	Unbenutzte Volumes löschen
`docker network create NAME`	Netzwerk erstellen
`docker network ls`	Netzwerke auflisten
`docker network rm NAME`	Netzwerk löschen
`docker network inspect NAME`	Netzwerk-Details
`docker network connect NET CONT`	Container zu Netzwerk hinzufügen

docker run Optionen (wichtigste)

Option	Bedeutung
`-d` / `--detach`	Im Hintergrund starten
`-it`	Interaktives Terminal (-i: stdin offen, -t: TTY)
`--name NAME`	Container benennen
`-p HOST:CONT`	Port-Weiterleitung (z.B. `-p 8080:80`)
`-v HOST:CONT`	Volume / Bind-Mount (-v /lokal:/container)
`--volume=vol:/pfad`	Named Volume einbinden
`-e VAR=WERT`	Umgebungsvariable setzen
`--env-file DATEI`	Umgebungsvariablen aus Datei
`--rm`	Container nach Beenden automatisch löschen
`--restart POLICY`	Neustart-Policy (no, always, on-failure, unless-stopped)
`--network NETZWERK`	Container in Netzwerk einbinden
`--network=host`	Host-Netzwerk direkt nutzen
`-u USER[:GRP]`	Als bestimmten Benutzer ausführen
`-w DIR`	Arbeitsverzeichnis im Container
`--entrypoint CMD`	Standard-Entrypoint überschreiben
`--memory=512m`	RAM-Limit setzen
`--cpus=1.5`	CPU-Limit setzen
`--read-only`	Dateisystem des Containers schreibgeschützt
`--tmpfs /tmp`	In-Memory-Dateisystem für bestimmten Pfad
`-l KEY=WERT`	Label setzen
`--hostname NAME`	Hostname im Container

Dockerfile Aufbau

```
# ── Basisimage ──────────────────────────────────────────────
FROM ubuntu:22.04
# oder für kleinere Images:
FROM alpine:3.19
FROM debian:bookworm-slim

# ── Metadaten ───────────────────────────────────────────────
LABEL maintainer="admin@example.com"
LABEL version="1.0"
LABEL description="Meine Anwendung"

# ── Umgebungsvariablen ──────────────────────────────────────
ENV APP_VERSION=1.0 \
    NODE_ENV=production \
    PORT=3000

# ── Build-Argumente (nur zur Build-Zeit verfügbar) ──────────
ARG BUILD_DATE
ARG GIT_COMMIT

# ── Benutzer und Arbeitsverzeichnis ─────────────────────────
RUN groupadd -r appuser && useradd -r -g appuser appuser
WORKDIR /app

# ── Pakete installieren ─────────────────────────────────────
RUN apt-get update && apt-get install -y \
        curl \
        nginx \
        && rm -rf /var/lib/apt/lists/*    # Image-Größe reduzieren!

# ── Dateien kopieren ────────────────────────────────────────
COPY . . # Alles aus Build-Kontext kopieren
COPY package.json package-lock.json ./ # Zuerst nur package-Dateien
RUN npm ci --only=production      # Cache-freundlich: nur bei Paket-
Änderung
COPY src/ ./src/ # Dann den Rest

ADD archiv.tar.gz /app/           # ADD kann auch Archive entpacken

# ── Port deklarieren (Dokumentation, keine echte Weiterleitung) –
EXPOSE 3000
EXPOSE 80/tcp
EXPOSE 53/udp

# ── Volume deklarieren ──────────────────────────────────────
VOLUME ["/app/data", "/app/logs"]

# ── Gesundheitscheck ────────────────────────────────────────
HEALTHCHECK --interval=30s --timeout=10s --retries=3 \
      CMD curl -f http://localhost:3000/health || exit 1

# ── Benutzer wechseln (nie als root laufen!) ────────────────
USER appuser

# ── Entrypoint vs CMD ───────────────────────────────────────
# ENTRYPOINT: unveränderlicher Hauptbefehl
# CMD: Standardargumente (überschreibbar mit docker run ... BEFEHL)
ENTRYPOINT ["node"]
CMD ["server.js"]
# → docker run image              → node server.js
# → docker run image app.js       → node app.js
# → docker run --entrypoint sh image  → sh
```

```
# Nur CMD (kein Entrypoint):
CMD ["nginx", "-g", "daemon off;"]
```

Beispiele

```
# Einfacher Container-Start
docker run hello-world

# Nginx-Webserver auf Port 8080
docker run -d -p 8080:80 --name mein-nginx nginx:alpine

# Interaktive Shell in Ubuntu
docker run -it --rm ubuntu:22.04 bash

# Image bauen
docker build -t meine-app:1.0 .
docker build -t meine-app:1.0 -f Dockerfile.prod .

# Container-Logs verfolgen
docker logs -f mein-container

# In laufenden Container einsteigen
docker exec -it mein-container bash
docker exec -it mein-container sh # für Alpine-Images

# Port-Info anzeigen
docker port mein-container

# Alle Container und Images aufräumen
docker system prune -af --volumes

# Ressourcen aller laufenden Container
docker stats --no-stream

# Container-IP-Adresse
docker inspect mein-container | grep -i ipaddress
```

docker compose

Beschreibung: `docker compose` (früher `docker-compose`) verwaltet **Multi-Container-Anwendungen** über eine einzige YAML-Datei. Es orchestriert das gemeinsame Starten, Stoppen und Konfigurieren mehrerer zusammengehöriger Container.

Syntax: `docker compose [OPTIONEN] BEFEHL [SERVICE...]`
Standarddatei: `docker-compose.yml` (oder `docker-compose.yaml`)

Alle docker compose Befehle

Befehl	Bedeutung
`up`	Container erstellen und starten
`down`	Container stoppen und entfernen
`start`	Gestoppte Container starten (ohne neu zu erstellen)
`stop`	Container graceful stoppen (SIGTERM)
`restart`	Container neu starten
`pause`	Container einfrieren (SIGSTOP)
`unpause`	Eingefrorene Container fortsetzen
`build`	Images (neu) bauen
`pull`	Images aller Services herunterladen
`push`	Images in Registry hochladen
`ps`	Status aller Container
`logs`	Logs aller oder bestimmter Services
`exec`	Befehl in laufendem Container ausführen
`run`	Einmaligen Befehl in neuem Container ausführen
`config`	Konfiguration ausgeben / validieren
`images`	Images der definierten Services auflisten
`rm`	Gestoppte Container entfernen
`kill`	Container sofort mit Signal beenden
`top`	Prozesse in laufenden Containern
`events`	Echtzeit-Ereignisstream ausgeben
`port`	Published Port eines Services anzeigen
`cp`	Dateien zwischen Host und Container kopieren
`wait`	Warten bis Container beendet sind
`watch`	Quellcode-Änderungen live in Container übertragen
`version`	Versionsinformationen

Optionen für `docker compose up`

Option	Bedeutung
`-d` / `--detach`	Im Hintergrund starten (detached mode)
`--build`	Images neu bauen vor dem Start
`--no-build`	Images nicht bauen (auch wenn Dockerfile neuer)
`--no-recreate`	Existierende Container nicht neu erstellen
`--force-recreate`	Container immer neu erstellen
`--always-recreate-deps`	Abhängigkeiten immer neu erstellen
`--remove-orphans`	Container für nicht mehr definierte Services entfernen
`--scale SERVICE=N`	Service auf N Instanzen skalieren
`-t N` / `--timeout N`	Stop-Timeout in Sekunden (Standard: 10)
`--wait`	Warten bis alle Container `healthy` sind
`--wait-timeout N`	Timeout für `--wait`
`--no-attach SERVICE`	Logs dieses Services nicht anzeigen
`--pull always/missing/never`	Pull-Policy für Images

Optionen für `docker compose down`

Option	Bedeutung
`-v` / `--volumes`	Named Volumes und anonyme Volumes ebenfalls löschen
`--rmi all`	Alle Images der Services entfernen
`--rmi local`	Nur lokal gebaute Images entfernen
`--remove-orphans`	Orphan-Container entfernen
`-t N` / `--timeout N`	Stop-Timeout

Optionen für `docker compose logs`

Option	Bedeutung
`-f` / `--follow`	Logs live verfolgen
`--tail=N`	Nur letzte N Zeilen anzeigen
`-t` / `--timestamps`	Zeitstempel voranstellen
`--no-log-prefix`	Service-Namen nicht voranstellen
`--since ZEIT`	Logs ab bestimmtem Zeitpunkt (z.B. `1h`, `2024-01-01`)
`--until ZEIT`	Logs bis zu bestimmtem Zeitpunkt

Optionen für `docker compose exec`

Option	Bedeutung
`-it`	Interaktives Terminal
`-u USER`	Als Benutzer ausführen
`-e VAR=WERT`	Umgebungsvariable setzen
`-w DIR`	Arbeitsverzeichnis setzen
`--no-TTY`	Kein TTY
`--index=N`	Bei skaliertem Service: N-te Instanz (Standard: 1)

Allgemeine Optionen (vor dem Befehl)

Option	Bedeutung
`-f DATEI`	Andere Compose-Datei verwenden
`-p NAME`	Projektname festlegen (Standard: Verzeichnisname)
`--profile PROFIL`	Service-Profile aktivieren
`--env-file DATEI`	Andere `.env`-Datei verwenden
`--no-ansi`	Keine Farb-Ausgabe
`--progress auto/tty/plain/quiet`	Ausgabe-Stil
`--ansi auto/never/always`	ANSI-Farben steuern

Aufbau einer docker-compose.yml – vollständiges Beispiel

```yaml
version: '3.9'
services:
  frontend:
    build:
      context: ./client-react
      dockerfile: Dockerfile
      args:
        - REACT_APP_VERSION=1.0.0
    image: 'mein-frontend:latest'
    container_name: frontend
    hostname: frontend
    restart: unless-stopped
    ports:
      - '3000:3000'
      - '127.0.0.1:4000:4000'
    volumes:
      - './src:/app/src'
      - /app/node_modules
    environment:
      NODE_ENV: development
      API_URL: 'http://backend:8080'
    env_file:
      - .env
      - .env.local
    networks:
      - frontend-net
    depends_on:
      backend:
        condition: service_healthy
    healthcheck:
      test:
        - CMD
        - curl
        - '-f'
        - 'http://localhost:3000'
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 60s
    logging:
      driver: json-file
      options:
        max-size: 10m
        max-file: '3'
    labels:
      - traefik.enable=true
      - traefik.http.routers.frontend.rule=Host(`app.example.com`)
    deploy:
      resources:
        limits:
          cpus: '0.5'
          memory: 256M
        reservations:
          memory: 128M
  backend:
    build:
      context: ./api-golang
    restart: always
    ports:
      - '8080:8080'
    environment:
      - DB_HOST=database
```

```
      - DB_PORT=5432
      - 'DB_NAME=${POSTGRES_DB}'
      - 'DB_USER=${POSTGRES_USER}'
      - 'DB_PASSWORD=${POSTGRES_PASSWORD}'
    networks:
      - frontend-net
      - backend-net
    depends_on:
      database:
        condition: service_healthy
    healthcheck:
      test:
        - CMD-SHELL
        - 'wget -q --spider http://localhost:8080/health || exit 1'
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 30s
  database:
    image: 'postgres:15-alpine'
    restart: always
    volumes:
      - 'db_data:/var/lib/postgresql/data'
      - './init.sql:/docker-entrypoint-initdb.d/init.sql'
    environment:
      POSTGRES_DB: '${POSTGRES_DB:-myapp}'
      POSTGRES_USER: '${POSTGRES_USER:-user}'
      POSTGRES_PASSWORD: '${POSTGRES_PASSWORD:?DB-Passwort
erforderlich!}'
    networks:
      - backend-net
    healthcheck:
      test:
        - CMD-SHELL
        - 'pg_isready -U ${POSTGRES_USER:-user}'
      interval: 10s
      timeout: 5s
      retries: 5
    expose:
      - '5432'
  cache:
    image: 'redis:7-alpine'
    restart: unless-stopped
    command: redis-server --appendonly yes
    volumes:
      - 'redis_data:/data'
    networks:
      - backend-net
    profiles:
      - cache
volumes:
  db_data:
    driver: local
  redis_data: null
  nginx_logs:
    driver: local
    driver_opts:
      type: none
      o: bind
      device: /srv/logs/nginx
networks:
  frontend-net:
    driver: bridge
  backend-net:
```

```
    driver: bridge
    internal: true
  monitoring:
    external: true
    name: monitoring_net
```

restart-Policies

Wert	Bedeutung
`no`	Niemals neu starten (Standard)
`always`	Immer neu starten – auch nach manuellem Stopp
`on-failure`	Nur bei Fehler (Exit-Code ≠ 0) neu starten
`unless-stopped`	Immer neu starten, außer wenn manuell gestoppt

depends_on – condition-Werte

Wert	Bedeutung
`service_started`	Warten bis der Container gestartet ist (Standard)
`service_healthy`	Warten bis der Container `healthy` ist (Healthcheck nötig!)
`service_completed_successfully`	Warten bis der Container mit Exit 0 beendet wurde

.env-Datei

```
# .env (im selben Verzeichnis wie docker-compose.yml)
# Automatisch von docker compose geladen

POSTGRES_DB=myapp
POSTGRES_USER=app_user
POSTGRES_PASSWORD=sicheres_passwort_123
IMAGE_TAG=1.5.2
APP_PORT=3000
```

Verwendung im docker-compose.yml:

```
services:
  app:
    image: 'meineapp:${IMAGE_TAG:-latest}'
    ports:
      - '${APP_PORT}:3000'
    environment:
      DB_PASS: '${POSTGRES_PASSWORD:?Passwort muss gesetzt sein!}'
```

Praktische Beispiele

```
# ── Starten ──────────────────────────────────────────────────
# Alle Services im Hintergrund starten
docker compose up -d

# Starten und Live-Logs beobachten (kein -d)
docker compose up

# Images neu bauen und dann starten
docker compose up -d --build

# Nur bestimmte Services starten
docker compose up -d database cache

# Mit einem Profil starten
docker compose --profile cache up -d

# Warten bis alle Container healthy sind
docker compose up -d --wait

# ── Logs ─────────────────────────────────────────────────────
# Alle Logs live verfolgen
docker compose logs -f

# Nur Backend-Logs, letzte 100 Zeilen
docker compose logs --tail=100 -f backend

# Logs mit Zeitstempel
docker compose logs -t

# ── Befehle ausführen ────────────────────────────────────────
# Bash im Backend-Container
docker compose exec backend bash

# Datenbankzugriff
docker compose exec database psql -U app_user -d myapp

# HTTP-Anfrage aus dem Container
docker compose exec backend curl http://database:5432

# Einmaliger Befehl (neuer Container, wird danach gelöscht)
docker compose run --rm backend npm test
docker compose run --rm backend python manage.py migrate

# ── Status und Diagnose ──────────────────────────────────────
# Status aller Container
docker compose ps

# Mit Ports und Healthcheck
docker compose ps -a

# Ressourcenverbrauch
docker stats $(docker compose ps -q)

# Prozesse in Containern
docker compose top

# Konfiguration validieren und anzeigen
docker compose config

# ── Stoppen und Aufräumen ────────────────────────────────────
# Stoppen (Container bleiben erhalten)
docker compose stop
```

```
# Stoppen und Container entfernen
docker compose down

# Stoppen, Container UND Volumes entfernen (VORSICHT: Datenverlust!)
docker compose down -v

# Stoppen und auch Images entfernen
docker compose down --rmi all

# Nur bestimmte Services neu starten
docker compose restart backend

# ── Skalieren ──────────────────────────────────────────────
# Backend auf 3 Instanzen skalieren
docker compose up -d --scale backend=3

# ── Mehrere Compose-Dateien ────────────────────────────────
# Produktion mit Überschreibungs-Datei
docker compose -f docker-compose.yml -f docker-compose.prod.yml up -
d

# Mit eigenem Projektname
docker compose -p mein-projekt up -d

# Mit anderer .env-Datei
docker compose --env-file .env.production up -d
```

Ausgabe

```
    # docker compose ps
    NAME                    IMAGE               COMMAND                  SERVICE    CREATED         STATUS
PORTS
    mein-projekt-backend-1  backend:latest      "./server"               backend    2 hours ago     Up 2 hours
(healthy)     0.0.0.0:8080->8080/tcp
    mein-projekt-frontend-1 frontend:latest     "nginx -g 'daemon of…"  frontend   2 hours ago     Up 2 hours
0.0.0.0:3000->3000/tcp
    mein-projekt-database-1 postgres:15-alpine  "docker-entrypoint.s…"  database   2 hours ago     Up 2 hours (healthy)
```

Best Practice: Niemals Passwörter direkt in `docker-compose.yml` schreiben – immer `.env`-Datei verwenden und `.env` in `.gitignore` eintragen!

/etc/shadow

Beschreibung: Enthält die **verschlüsselten Passwörter** und Passwort-Ablaufinformationen aller Systembenutzer. Nur **root** (und die Gruppe **shadow**) kann diese Datei lesen.

Berechtigungen:

-rw-r----- 1 root shadow <- auf Debian/Ubuntu
-rw------- 1 root root <- auf RHEL/CentOS

Format (9 Felder, Trenner: :)

benutzername:passwort-hash:letzte-änderung:min-tage:max-tage:warn-tage:inaktiv-tage:ablauf:reserviert

Feldübersicht

Feld-Nr.	Name	Beschreibung
1	benutzername	Login-Name (muss mit /etc/passwd übereinstimmen)
2	passwort-hash	Gehashtes Passwort mit Algorithmus-Kennung
3	letzte-änderung	Tage seit 01.01.1970 der letzten Passwortänderung. 0 = muss bei nächstem Login geändert werden
4	min-tage	Minimum-Tage bis Passwort geändert werden darf (0 = jederzeit)
5	max-tage	Maximum-Gültigkeit in Tagen (99999 = läuft nie ab)
6	warn-tage	Warnung N Tage vor Ablauf (leer = keine Warnung)
7	inaktiv-tage	Tage nach Ablauf bis Konto deaktiviert wird (leer = nie)
8	ablauf	Absolutes Ablaufdatum (Tage seit 01.01.1970, leer = nie)
9	reserviert	Für zukünftige Verwendung reserviert, immer leer

Passwort-Hash Formate

Präfix	Algorithmus	Sicherheit
1	MD5	Veraltet, unsicher!
$2a$ / $2b$	bcrypt	Gut
5	SHA-256	Gut
6	SHA-512	Empfohlen (Standard auf vielen Systemen)
y	yescrypt	Modern, sehr sicher (Standard Debian 11+)
7	scrypt	Modern, sicher
!	–	Konto gesperrt (passwd -l), ! vor dem Hash
!!	–	Konto gesperrt und nie ein Passwort gesetzt
*	–	Kein Login möglich (Systemkonten wie daemon, bin)
leer	–	Kein Passwort – Login ohne Passwort möglich (sehr unsicher!)

SHA-512-Hash-Struktur

```
$6$rounds=65536$saltwert$eigentlicher_hash
| |       |     |
| |       |     +-- Base64-kodierter Hash (86 Zeichen)
| |       +-- Salt (zufälliger Wert, bis 16 Zeichen)
| +-- rounds: Anzahl der Iterationen (Standard: 5000)
+-- Algorithmus: 6 = SHA-512
```

Vollständige Beispielzeilen

```
# Normales Konto mit SHA-512-Passwort
alice:$6$rounds=65536$abc123xyz$HashHashHash...:19845:0:99999:7:::
|     |                          |               |     | |     |
|     |                          |               |     | |     +-- 7
Tage Warnung
|     |                          |               |     | +-- max
99999 Tage
|     |                          |               |     +-- min 0
Tage
|     |                          |               +-- letzte Änderung
|     |                          +-- Hash
|     +-- $6$ = SHA-512
+-- Benutzername

# Gesperrtes Konto
bob:!$6$hash...:19845:0:99999:7:::
    ^-- ! = gesperrt

# Noch nie Passwort gesetzt
newuser:!!:19845:0:99999:7:::
        ^^-- !! = kein Passwort, gesperrt

# Systemkonto
daemon:*:19845:0:99999:7:::
       ^-- * = kein Login möglich
```

Nützliche Befehle

```
# Anzeigen (nur als root)
sudo cat /etc/shadow

# Einzelnen Benutzer abfragen
sudo getent shadow alice

# Passwortstatus anzeigen
sudo passwd -S alice

# Ausführliche Ablaufinformationen
sudo chage -l alice

# Feld 3 (Tage) in Datum umrechnen
date -d "1970-01-01 + 19845 days"

# Konten ohne Passwort finden (Sicherheitscheck!)
sudo awk -F: '$2 == "" {print $1}' /etc/shadow

# Gesperrte Konten finden
sudo awk -F: '$2 ~ /^!/ {print $1}' /etc/shadow

# Konten mit niemals ablaufendem Passwort
sudo awk -F: '$5 == 99999 {print $1}' /etc/shadow
```

/etc/group

Beschreibung: Enthält alle Gruppen des Systems mit ihren GIDs und Mitgliedern. Jeder Benutzer hat eine **primäre Gruppe** (in **/etc/passwd** gespeichert) sowie beliebig viele **sekundäre Gruppen** (hier eingetragen).

Berechtigungen: -rw-r--r-- 1 root root (von allen lesbar)

Format (4 Felder, Trenner: :)

gruppenname:passwort:GID:mitglieder

Feldübersicht

Feld-Nr.	Name	Beschreibung
1	gruppenname	Name der Gruppe
2	passwort	x = Passwort in /etc/gshadow; leer = kein Passwort
3	GID	Gruppen-ID (numerisch). 0 = root, 1-999 = System, 1000+ = normal
4	mitglieder	Kommagetrennte Liste der sekundären Mitglieder

GID-Bereiche (Konvention)

Bereich	Verwendung
0	root-Gruppe
1 – 99	Statische System-Gruppen
100 – 999	Dynamische System-/Dienst-Gruppen
1000+	Normale Benutzergruppen

Vollständiges Beispiel

```
root:x:0:
daemon:x:1:
adm:x:4:syslog,matta
sudo:x:27:matta,alice
cdrom:x:24:matta
plugdev:x:46:matta
docker:x:999:matta,alice
entwickler:x:1001:alice,bob,charlie
webteam:x:1002:bob,diana
```

Unterschied: Primäre vs. Sekundäre Gruppe

```
# /etc/passwd
alice:x:1001:1001:Alice:/home/alice:/bin/bash
              ^^^^ GID 1001 = primäre Gruppe von alice

# /etc/group
entwickler:x:1001:alice,bob
# alice taucht hier als sekundäres Mitglied auf
# Würde alice diese Gruppe als primäre haben, stünde sie NICHT hier
```

Nützliche Befehle

```
# Datei anzeigen
cat /etc/group

# Alle Gruppen des aktuellen Benutzers
groups

# Alle Gruppen eines bestimmten Benutzers
groups alice
id -Gn alice

# Alle GIDs numerisch
id -G alice

# Gruppeninformationen abfragen
getent group entwickler
getent group 1001

# Benutzer zu Gruppe hinzufügen (gilt ab nächstem Login)
sudo usermod -aG entwickler alice
# WICHTIG: ohne -a werden alle bisherigen Gruppen ersetzt!

# Benutzer aus Gruppe entfernen
sudo gpasswd -d alice entwickler

# Neue Gruppe erstellen
sudo groupadd projektteam

# Neue Gruppe mit spezifischer GID
sudo groupadd -g 2000 ops

# Gruppe umbenennen
sudo groupmod -n neuername altername

# Gruppe löschen
sudo groupdel altername

# Alle Mitglieder der sudo-Gruppe
getent group sudo | cut -d: -f4
```

Ausgabe

```
# getent group entwickler
entwickler:x:1001:alice,bob,charlie
#          | |    +-- sekundäre Mitglieder (kommagetrennt)
#          | +-- GID
#          +-- x = Passwort in /etc/gshadow

# groups alice
alice : alice adm sudo entwickler docker
#       ^^^^^ primäre Gruppe
#             ^^^^^^^^^^^^^^^^^^^^^^^^^ sekundäre Gruppen
```

/etc/gshadow

Beschreibung: Enthält verschlüsselte **Gruppenpasswörter** sowie Informationen zu Gruppenadministratoren. Nur **root** und die Gruppe **shadow** können diese Datei lesen.

Berechtigungen: -rw-r----- 1 root shadow

Format (4 Felder, Trenner: :)

gruppenname:passwort-hash:administratoren:mitglieder

Feldübersicht

Feld-Nr.	Name	Beschreibung
1	gruppenname	Name der Gruppe (muss mit /etc/group übereinstimmen)
2	passwort-hash	Gehashtes Gruppenpasswort. ! / !! = kein Passwort / gesperrt; leer = kein Passwort
3	administratoren	Kommagetrennte Liste von Gruppenadmins – können Mitglieder hinzufügen/entfernen ohne root zu sein
4	mitglieder	Kommagetrennte Liste der Mitglieder (spiegelt /etc/group Feld 4)

Gruppenpasswort – Wofür?

Gruppenpasswörter erlauben es einem Benutzer, der **nicht Mitglied** einer Gruppe ist, mittels **newgrp <gruppe>** temporär die Gruppe zu wechseln – wenn er das Gruppenpasswort kennt.

```
# Gruppenpasswort setzen
sudo gpasswd entwickler

# Nicht-Mitglied kann beitreten wenn er das Passwort kennt
newgrp entwickler
# Passwort:
```

Vollständiger Beispielinhalt

```
root:::
daemon:::
sudo:!:root:matta,alice
entwickler:$6$AbcDef...:alice:alice,bob,charlie
#          |              |    +-- normale Mitglieder
#          |              +-- alice ist Gruppenadmin
#          +-- gehashtes Gruppenpasswort
webteam:!!::bob,diana
#       ^^-- kein Passwort + gesperrt
```

Nützliche Befehle

```
# Anzeigen (nur als root)
sudo cat /etc/gshadow

# Einzelnen Eintrag abfragen
sudo getent gshadow entwickler

# Gruppenpasswort setzen
sudo gpasswd entwickler

# Gruppenpasswort entfernen (jeder kann beitreten)
sudo gpasswd -r entwickler

# Gruppenadmin hinzufügen
sudo gpasswd -A alice entwickler

# Mitglied hinzufügen (als Gruppenadmin oder root)
sudo gpasswd -a bob entwickler

# Mitglied entfernen
sudo gpasswd -d charlie entwickler

# Mitglieder komplett ersetzen
sudo gpasswd -M alice,bob entwickler

# Als Gruppenadmin die Gruppe temporär wechseln
newgrp entwickler
```

Ausgabe

```
# sudo getent gshadow entwickler
entwickler:$6$AbcDef...:alice:alice,bob,charlie
#          |             |    +-- Mitglieder
#          |             +-- Admin: alice
#          +-- Hash des Gruppenpassworts

# sudo gpasswd -a diana entwickler
Benutzer diana wurde zur Gruppe entwickler hinzugefügt.
```

Schnellreferenz: Berechtigungen

Vollständige Oktal-Tabelle

Oktal	Symbolisch	Beschreibung
777	rwxrwxrwx	Alle Rechte für alle (gefährlich!)
755	rwxr-xr-x	Standard für Verzeichnisse und Skripte
750	rwxr-x---	Gruppe kann lesen, Andere nichts
700	rwx------	Nur Eigentümer darf alles
664	rw-rw-r--	Kollaborativer Dateizugriff
644	rw-r--r--	Standard für Dateien
640	rw-r-----	Sensible Konfigdateien
600	rw-------	Private Schlüssel (z.B. ~/.ssh/id_rsa)
444	r--r--r--	Nur-Lesen für alle
400	r--------	Nur Eigentümer kann lesen
4755	rwsr-xr-x	SUID gesetzt
4750	rwsr-x---	SUID + Gruppe kann ausführen
2775	rwxrwsr-x	SGID auf Verzeichnis (Gruppe vererbt sich)
1777	rwxrwxrwt	Sticky Bit (wie /tmp)
1755	rwxr-xr-t	Sticky Bit + normaler Zugriff

Rechte-Struktur

```
Typ  Eigentümer  Gruppe  Andere
-   r w x     r w x   r w x
    4 2 1     4 2 1   4 2 1
```

Dateitypen in ls-l

Zeichen	Typ
-	Reguläre Datei
d	Verzeichnis (directory)
l	Symbolischer Link
c	Zeichengerät (character device)
b	Blockgerät (block device)
p	Named Pipe (FIFO)
s	Unix-Socket

Spezialbit-Anzeige in ls -l

Position	Gesetzt + x	Gesetzt, kein x	Bit
Eigentümer-x	s	S	SUID
Gruppen-x	s	S	SGID
Andere-x	t	T	Sticky

Wichtige Systemdateien und ihre Rechte

Datei	Rechte	Eigentümer	Warum
/etc/passwd	644	root:root	Jeder kann lesen
/etc/shadow	640	root:shadow	Nur root/shadow
/etc/group	644	root:root	Jeder kann lesen
/etc/gshadow	640	root:shadow	Nur root/shadow
/etc/sudoers	440	root:root	Nur Lesen
~/.ssh/	700	user:user	Nur Eigentümer
~/.ssh/id_rsa	600	user:user	Nur Eigentümer
~/.ssh/authorized_keys	600	user:user	Nur Eigentümer
/tmp	1777	root:root	Sticky Bit
/usr/bin/passwd	4755	root:root	SUID

umask-Berechnung

Neue Datei: 666
minus umask: -022

Ergebnis: 644 (rw-r--r--)

Neues Verzeichnis: 777
minus umask: -022

Ergebnis: 755 (rwxr-xr-x)