Linux Commands

Linux Commands

Basics

Ctrl+Shift+T	Terminal Öffnen	**Steuerung Terminal**
Ctrl+C	Abrechen / neue Befehlszeile	
Ctrl+D	Log out	
Ctrl+L	Verlauf löschen	
cd ..	Ordner zurück ⬅ □□	
cd ~	in home wechseln ➡ □□	
touch	Datei erstellen □ □□	
mkdir	Ordner erstellen □ □□	
rm -r	Löschen □ □□	
ls	Liste anzeigen □□+□□	□□ **Anzeigen mit LS**
ls -l /home	anzeigen aller /home Ordner /□□ /□□	
ls -l	Inhalt anzeigen. □□ □□ □□ □□	
ls -la (optional -h = KB, MB)	Inhalt anzeigen. (Rechte,User,Group,Größe,Datum+versteckte) . □□ □□ □□ □□ □□	

tree	tree struktur anzeigen □□└──	□□ **Anzeigen Struktur**
tree -L 1	Zeigt erstes unter level an □□├──	
tree -pug	tree mit □□ -pug(**Zugriffsrecht, User, Gruppe**) □□,□□,□□	
getent passwd	Alle user anzeigen □□□□□□	□□ **Anzeigen User & Group**
getent group	Gruppe anzeigen □□□□□□	
# getent shadow	Passwort-Hashes anzeigen Sudo oder aus Root #□ □□	
# useradd -m -s /bin/bash user	Benutzer erstellen + Home + Bash. □ □□□□ & (erstellt Standart Hauptgruppe mit user Namen)	**Benutzer &** **Gruppen erstellen**
# useradd -m -s /bin/bash -g HAUPT -G NEBENGRUPPE user	Benutzer erstellen + Home + Bash □ □□□□ +HauptG +NebenG □□ (□□)	-g Hauptgruppe -G Nebengruppe (alleinstehend ersetzt alle bisherigen nebengrupen) -aG Nebengruppe adden
# groupadd groupName	neue Gruppe erstellen □□ □	

# usermod -aG NebenGruppe user	User zur Nebengruppe hinzufügen. □□→□□	
# groupmod -n neuerName alterName	Gruppe umbenennen. □□ □□Name	
# chgrp -R gruppe /...	Gruppe ändern + ALLE Daten □□ + □□	**-R = Rekursiv (Alle Dateien)**
# usermod -s /bin/bash user	Bash nachträglich hinzufügen	
{ }	Kaskadieren	
cp	Kopie	**kopieren oder verschieben** Bsp. mv OldName NewName
mv	Verschiebt o. benennt Dateien	
getent passwd	Alle user anzeigen	
getent group	Gruppe anzeigen	
# passwd user	Passwort zuweisen	
su - user	Nutzer wechseln	
# chown user:group /home/OldUser	Besitzer übergeben □□ □□□□ □□	

# chgrp group datei	Ändert nur gruppe □□... □□ → □□	
# chown user:group dateiname	Ändert besitzer & gruppe. □□... □□□□ → □□	
# usermod -l oldUser newUser	Ändern des namen □□□□Name	
fdisk -L	Zeigt alle festplatten und formate an	
crontab -e	Um zeitlich gesteuerte dinge zu starten	

= sudo

CHMOD EINSTELLUNGEN:

- **Löschen hängt vom Ordner ab, nicht von der Datei**
  → benötigt: `w + x` auf dem Ordner
- **Ordnerrechte ≠ Dateirechte**
  → Datei: `r=lesen`, `w=ändern`, `x=ausführen`
  → Ordner: `r=sehen`, `w=erstellen/löschen`, `x=betreten`
- **`x` ist Pflicht für Zugriff**
  → ohne `x`: kein `cd`, kein Öffnen, egal was sonst gesetzt ist
- **`r` ohne `x`**
  → sehen eingeschränkt möglich, aber nicht nutzen
- **`x` ohne `r`**
  → betreten möglich, aber kein `ls` (Blindzugriff)
- **`w` ohne `x`**
  → praktisch nutzlos (kein Zugriff auf den Ordner)
- **Verschieben (`mv`)**
  → benötigt `w + x` auf Quell- UND Zielordner
- **Datei löschen ohne Dateirechte möglich**
  → wenn Ordner `w + x` hat
- **root ignoriert Rechte weitgehend**
  → kann fast alles unabhängig von Permissions
- **`chmod 777`**
  → funktioniert immer, aber unsicher (jeder darf alles)
- **Sticky Bit (`t`) z. B. `/tmp`**
  → alle dürfen schreiben, aber nur eigene Dateien löschen

--

Linux Prompt (Befehlszeile) – komplett kompakt

Grundaufbau

`user@hostname:pfad$`
oder als Admin:
`root@hostname:pfad#`

Bestandteile

- `user` → aktueller Benutzer
  - `user` = normaler Benutzer
  - `root` = Administrator
- `hostname` → Rechner- oder Systemname
  - z. B. `server`, `NASSI`, `raspberrypi`, `container`
- `pfad` → aktueller Ordner
  - `~` = Home-Verzeichnis von `user` = `/home/user`
  - `/` = Root-Verzeichnis, also die oberste Ebene des Systems
  - `/etc`, `/var/log`, `/tmp` = absolute Pfade
  - `..` = eine Ebene zurück
- `$` / `#` → Rechte-Stufe
  - `$` = normaler Benutzer
  - `#` = root / Admin

Beispiele

`user@server:~$`
→ Benutzer `user`, Host `server`, im Home-Verzeichnis, normale Rechte

`root@server:~#`
→ Benutzer `root`, Host `server`, im Home-Verzeichnis von root (`/root`), volle Rechte

`user@server:/home/user/projects$`
→ Benutzer `user`, Host `server`, im Ordner `/home/user/projects`, normale Rechte

`root@server:/etc#`
→ Benutzer `root`, Host `server`, im Ordner `/etc`, volle Rechte

`user@NASSI:/var/log$`
→ Benutzer `user`, Host `NASSI`, im Log-Verzeichnis `/var/log`, normale Rechte

`root@container:/app#`
→ Benutzer `root`, Host `container`, im Ordner `/app`, volle Rechte

`user@raspberrypi:~$`
→ Benutzer `user`, Host `raspberrypi`, im Home-Verzeichnis, normale Rechte

`~/projects $`
→ verkürzter Prompt, nur Pfad sichtbar, aktueller Ordner ist `projects` im Home-Verzeichnis

`$`
→ minimaler Prompt, keine Infos zu Benutzer, Host oder Pfad sichtbar

`user@server:/tmp$`
→ Benutzer `user`, Host `server`, im temporären Ordner `/tmp`

`user@server:/$`
→ Benutzer `user`, Host `server`, im Root-Verzeichnis `/`

Wichtige Bedeutungen

- `~` = Home-Verzeichnis des aktuellen Benutzers
- `/` = oberste Ebene des gesamten Systems
- `..` = ein Verzeichnis zurück
- `user@host` = zeigt, **wer** du bist und **auf welchem System** du bist
- `pfad` = zeigt, **wo** du dich gerade befindest
- `$` oder `#` = zeigt, **mit welchen Rechten** du arbeitest

Typische Prüfungsfallen

- `#` bedeutet nicht Kommentar im Prompt, sondern **root-Rechte**
- `~` ist **nicht** dasselbe wie `/`
  - `~` = Benutzerbereich
  - `/` = gesamtes System
- Der Hostname ist wichtig, weil du damit erkennst, auf welchem Gerät du gerade arbeitest
- Ein Prompt mit `$` ist deutlich ungefährlicher als ein Prompt mit `#`

Merksatz

`user@host = WER und WO`
`pfad = an welcher Stelle im System`
`$ / # = wie viele Rechte du hast`