Linux OS

`I.BA_LIOS.H25` — Zusammenfassung

Author(s) Dominic, Hannah, Laura
Date 16. Mar. 2026
Pages 145

Inhaltsverzeichnis

1. Einführung

1.1. Woher stammt Linux?

- Linux ist über 30 Jahre alt
- begann als Hobby von Linus Benedict Torvalds
- Linux = Linus' Unix (ein OS, das von Unix-Systemen abstammt)
- Linus Ziel -> Unix auf einem Heim-PC laufen zu lassen

1.2. Was ist Linux?

DEFINITION: Linux ist eigentlich nur Betriebssystemkern (**Kernel**), der die Hardware für die einzelnen Anwendungen steuert und verwaltet.

1.2.1. Kernel

TL;DR: zentrale Teil des Betriebssystems

- Der **Kernel** ist der zentrale Teil eines Betriebssystems.
- steuert die **Hardware** (z. B. Prozessor, Speicher, Geräte) und sorgt dafür, dass Programme diese nutzen können.
- Aufgaben des Kernels sind zum Beispiel:
  - **Speicherverwaltung**: Wer darf welchen Arbeitsspeicher nutzen?
  - **Multitasking**: Mehrere Programme gleichzeitig ausführen.
  - **Geräteverwaltung**: Über sogenannte **Gerätetreiber** können Programme mit angeschlossenen oder eingebauten Geräten kommunizieren (z. B. Maus, Tastatur, Festplatte, virtuelle Geräte).
  - **Netzwerk- und Dateisystemmanagement**: Kommunikation über Netzwerke und Verwaltung von Dateien.

nur der Kernel allein wäre nutzlose -> ein Betriebssystem enthält eine Reihe von System-Programmen. Ein wichtiges Programm vom Betriebssystem Linux bspw. ist, bash, ein Befehlszeileninterpreter (Shell)

1.2.2. Bestandteile von Linux

Linux besteht aus (auch Linuxdistribution genannt):

- Kernel
- Systemprogramme
- Compiler und die Toolchain zur Softwareentwicklung
- Freie und / oder proprietäre Software für verschiedene Aufgaben
  - Befehlszeileninterpreter
  - Anwendungsprogramme

1.3. Warum Linux

- ist **weit verbreitetet**
- wenn man das Internet nutzt, interagiert man mit Linux-Systemen
- Grundlegende Technologie für Cloud, Microservices-Anwendungen, softwarebasierte Speichertechnologien oder big-data
- **Stabilität**: Linux ist ein sehr stabiles System
- **Schutz**: Jeder Benutzer hat seinen eigenen privaten Datenbereich
- **Freier Quellcode**: Die Quellen des Betriebssystems und der meisten Anwendungen sind offen
- **Sicherheit**: Durch die frei verfügbaren Quellen kann jeder Sicherheitslücken patchen
- **Einfache Verwaltung**: Standardschritte lassen sich mit Hilfe von Skripten sehr einfach automatisieren
- **Läuft auf fast allen gängigen Hardwareplattformen**: von Mikrocomputern wie Smartphones bis hin zu Großrechnern wie z. B. AS400 von IBM
- Linux ist eine **modulare Distribution**, bei der Komponenten einfach ausgetauscht und entfernt werden können

1.3.1. Vorteile von Linux

- **Open source Software:** Verbesserungen werden einfacher vorgenommen und Innovation wird beschleunigt.
- **Linux bietet eine Befehlszeilenschnittstelle (Command-Line Interface, CLI)** vereinfacht:
  - Automatisierung
  - Deployment und Provisioning
  - erleichtert lokale und Remote-Systemadministration (war von Anfang an in der Architektur, nicht wie bei anderen Betriebssystemen)
- **stabil**: Linux ist ein sehr stabiles System
- **modulares Betriebssystem**: Kernel ist monolithisch aufgebaut (stellt schnelle Kommunikation innerhalb des Kernels sicher), Linux-Distribution ist modular (mit Paketmanager können Komponenten beliebig hinzugefügt werden)

1.4. GNU

DEFINITION: GNU = „GNU's Not Unix“, ein freies, unixähnliches, vollständiges Betriebssystem

- Start 1983 durch Richard Stallmann
- Ziel: komplett freies Betriebssystem -> Unix begann stark beschränkende Lizenzen zu veröffentlichen und den Quellcode geheim zu halten

1.4.1. Verbindung zu Linux

- Der eigene GNU-Kernel **Hurd** war nie praxistauglich.
- Deshalb kombiniert man **GNU-Programme** mit dem **Linux-Kernel** ⟶ Ergebnis: **GNU/Linux**(umgangssprachlich „Linux“).
- Erste Linux-Distributionen funktionierten hauptsächlich durch die Kombination von **Linux-Kernel + GNU-Software**.

1.4.2. GNU/Linux-Namensstreit

- Diskussion zwischen der **Freie-Software-Bewegung** und der **Open-Source-Community**: Soll man das System „Linux“ oder „GNU/Linux“ nennen?
- Hintergrund: Ohne GNU-Programme wäre Linux-Kernel alleine nicht nutzbar.

1.4.3. GNU-Projekt & Freie Software

- Zentrales Prinzip: **freie Software**.
- „Frei“ bedeutet nicht unbedingt kostenlos, sondern steht für bestimmte **Freiheiten**:
  1. Freiheit, das Programm auszuführen, wie man möchte.
  2. Freiheit, den Quellcode zu untersuchen und anzupassen.
  3. Freiheit, das Programm weiterzugeben.
  4. Freiheit, Verbesserungen weiterzugeben.

1.5. Geschichtliche Entwicklung von OpenSource und Linux

1970: Unix erblickt das Licht der Welt bei AT&T (Ken Thompson und Dennis Ritchie).

1979: Das bislang frei zugängliche Unix wird zum proprietären AT&T Unix, der Quellcode steht auch für Universitäten praktisch nicht mehr kostenfrei zur Verfügung.

1983: Richard Stallman gründet das GNU-Projekt mit dem Ziel, ein freies Betriebssystem zu erschaffen.

1989: Richard Stallman schreibt die erste Version der GNU General Public License (GPL).

1991: Der Linux-Kernel wird am 25. August von Linus Benedict Torvalds öffentlich im Usenet angekündigt. Am 17. September folgt die erste öffentliche Version auf einem FTP-Server.

1992: Der Linux-Kernel wird unter der GNU GPL vertrieben und es entstehen die ersten freien Linux-Distributionen

1993: Die älteste heute noch existierende Linux-Distribution Slackware wird das erste Mal veröffentlicht. Gründung der bis heute größten Community-Distribution Debian.

1994: Es dauert noch bis März dieses Jahres, bis Torvalds alle Komponenten im Kernel für ausgereift und vollständig erachtet und Linux in der Version 1.0 veröffentlicht: erstmals netzwerkfähig, XFree86 grafische Benutzerschnittstelle (GUI). Red Hat und SuSE veröffentlichen die Version 1.0 ihrer Linux-Distributionen.

1996: Die Version 2.0 des Kernels wird veröffentlicht. Der Kernel kann nun mehrere Prozessoren gleichzeitig bedienen

1997: Verschiedene proprietäre Programme: Browser Netscape Navigator, Office-Suiten Applixware und StarOffice

1998: Unternehmen wie IBM, Compaq und Oracle kündigen ihre Unterstützung für Linux an

1999: Die 2.2er-Serie erscheint im Januar mit verbessertem Netzwerkcode und verbesserter SMP-Unterstützung.

2001: Die 2.4er-Serie wird im Januar freigegeben. Der Kernel unterstützt nun bis zu 64 Gigabyte Arbeitsspeicher, 64-Bit-Dateisysteme, USB und Journaling-Dateisystem.

2008: Google veröffentlicht die erste Version von Android, welches sich in den folgenden Jahren zum vorherrschenden Betriebssystem auf Smartphones entwickelt. Basis ist ein Linux-Kernel.

2016: Microsoft integriert ein optionales Windows-Subsystem für Linux in Windows 10

1.6. Freie Software „OpenSource“

DEFINITION: Open Source ist Software, deren Quellcode von jedermann verwendet, eingesehen und modifiziert werden kann.

- von Einzelpersonen aus altruistischen Motiven
- auch von Organisationen / Unternehmen um Entwicklungskosten zu teilen oder Marktanteil zu gewinnen
- Start der Open-Source-Bewegung als Softwareunternehmen anfingen, in Maschinencode konvertierte Programme an ihre Kunden zu verkaufen.
- Befürchtung eines Kontrollverlusts und mangelnde Einsicht in die Software
- 1980er Jahren entstand das GNU-Projekt

1.6.1. Die wichtigsten Open-Source-Lizenzen

GPL-Lizenz:

- Software jederzeit frei verwendbar (auch kommerziell)
- Änderungen müssen unter GPL veröffentlicht werden (Copyleft- Prinzip)
- GPL-Code darf nicht in andere proprietäre (geschlossene) Software integriert werden.

LGPL- oder Apache-Lizenz:

- Software jederzeit frei verwendbar (auch kommerziell)
- Änderungen und Erweiterungen können auch unter anderen Lizenzen erfolgen. Abschwächung des Copyleft- Prinzips.
- auch proprietäre Erweiterungen möglich

Creative Common License (CCL):

- verschiedene Freiheitsgrade
- „nd“ (no derivatives): darf verwendet, aber nicht verändert werden
- „nc“ (non-commercial): darf nur nicht-kommerziell genutzt werden

1.6.2. Vorteile von OpenSource

- Viele Personen oder Unternehmen können sich an der Entwicklung beteiligen.
- nicht von einem Hersteller abhängig
  - Änderungen selber vornehmen oder jemanden damit beauftragen
  - bei proprietärer Software nicht möglich, respektive Änderung muss beim Hersteller beantragt werden
- Nutzung unterliegt keinen oder nur wenigen Bedingungen
  - beliebig viele User
  - beliebige Zwecke
- Einsicht in Quellcode -> Mehr-Augen-Prinzip macht die Software stabiler und zuverlässiger
- Es ist schwerer, versteckte Funktionen (z. B. Spionage) einzubauen, weil jeder den Code prüfen kann.

1.7. Die Kathedrale und der Basar (Eric S. Raymond, 1997)

1.7.1. Überblick

- Essay über Open-Source-Software
- Vortrag: 4. Internationaler Linux-Kongress, 22. Mai 1997, Würzburg
- Vergleich zweier Entwicklungsmodelle: **Kathedrale** vs. **Basar**

1.7.2. Kathedralen-Modell

- Quellcode nur bei Releases verfügbar
- Entwicklung durch kleine, geschlossene Gruppe
- Quellcode oft Betriebsgeheimnis
- Hierarchische Organisation wie beim Bau einer Kathedrale
- Ziel: Fertiges Produkt nach Plan

1.7.3. Basar-Modell

- Quellcode jederzeit öffentlich zugänglich (Internet)
- Viele Entwickler können jederzeit beitragen
- Selbstorganisation statt Hierarchie
- Beispiel: Linux-Kernel (Linus Torvalds als Maintainer)
- Software nie „fertig“, sondern kontinuierlicher Prozess
- Softwareindustrie = Dienstleistungs- statt Fertigungsindustrie

1.8. Linux-Distribution

DEFINITION: Auswahl aufeinander abgestimmter Software um den Linux-Kernel

Üblicherweise wird der Begriff auf Zusammenstellungen begrenzt, die weitgehend linux-typisch aufgebaut sind, was beispielsweise auf Android nicht zutrifft.

Inhalt

- Linux-Kernel
- Proprietäre Programme
- Distributionsprogramme
- GNU-Shell-Dienstprogramme
- X-Server und Desktop-Umgebung
- Paketverwaltungssystem und Installationsprogramm
- Patches
- Compiler
- Oft: Support, Handbücher und Verwaltungstools

2. Befehlszeile

2.1. Definition und Begriffe

Terminal / Konsole in einem Computer: Eingabe und Ausgabegerät

Terminal **virtuell** (z.B. Terminal auf Desktop) oder **physisch** (z.B SSH)

Grafikmodus Terminal mit Maus und Bildschirm (mit **GUI**)
- Beispiel: Verbindung über **RDP**

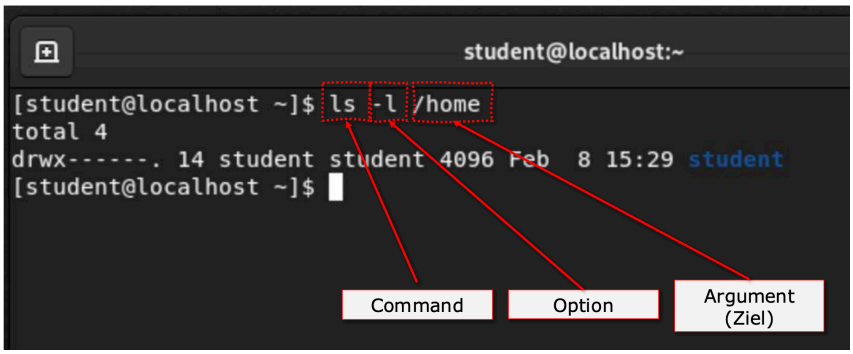
Textmodus Terminal im Textmodus - **command-line interface (CLI)**
- Beispiel: Verbindung über **SSH**

Kommandozeile / Befehlszeile / Eingabeaufforderung: Computerprogramm, das eine Textzeile als Eingabe vom Benutzer entgegennimmt und verarbeitet

Intepreter (Shell) ist Teil der Befehlszeile (somit auch Teil des gesamten Terminals)
- intepretiert die Eingabe (Befehl)
- zerlegt Eingabe
- Hilfe bei der Eingabe
- bezieht Kontext mit ein (aktuelles Verzeichnis)
- Bash, zsh, fish
- Benutzer können unterschiedliche Intepreter verwenden

Intepreter hat **Prompt / Command Prompt**
- visueller Anzeiger
- ist Teil der Befehlszeile / Kommandozeile / Eingabeaufforderung

Befehl besteht aus:
1. einem **Programm / Prompt**
2. mind. eine **Option** (Teil des Befehls)
3. mind. ein **Argument / Ziel** (nicht Teil des Befehls)



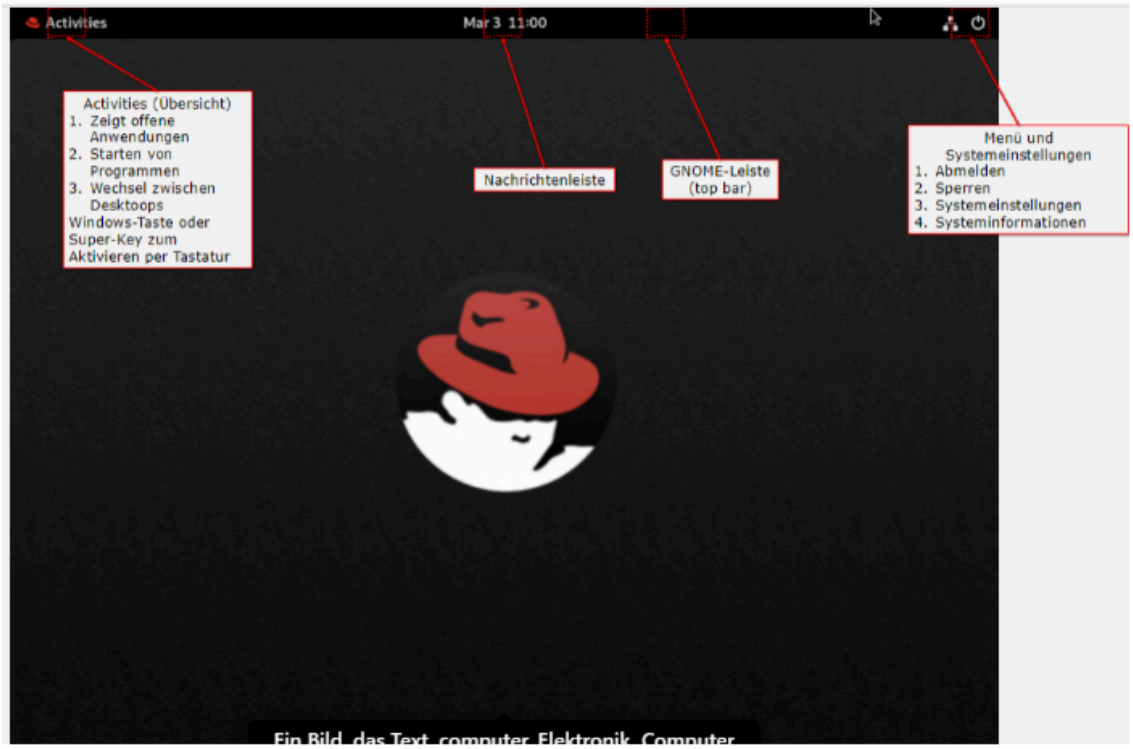
tty = Teletype, Teletypewriter
- Ursprung: Historische Fernschreibgeräte.
- Heutige Bedeutung: Eine textbasierte Schnittstelle zur Interaktion mit dem Betriebssystem.
- In Unix/Linux: Die Konsolen für Befehlseingaben.
- Besonderheit: Man kann oft via Tastenkombination (z. B. Strg + Alt + F1 bis F7) zwischen verschiedenen dieser Terminals wechseln.

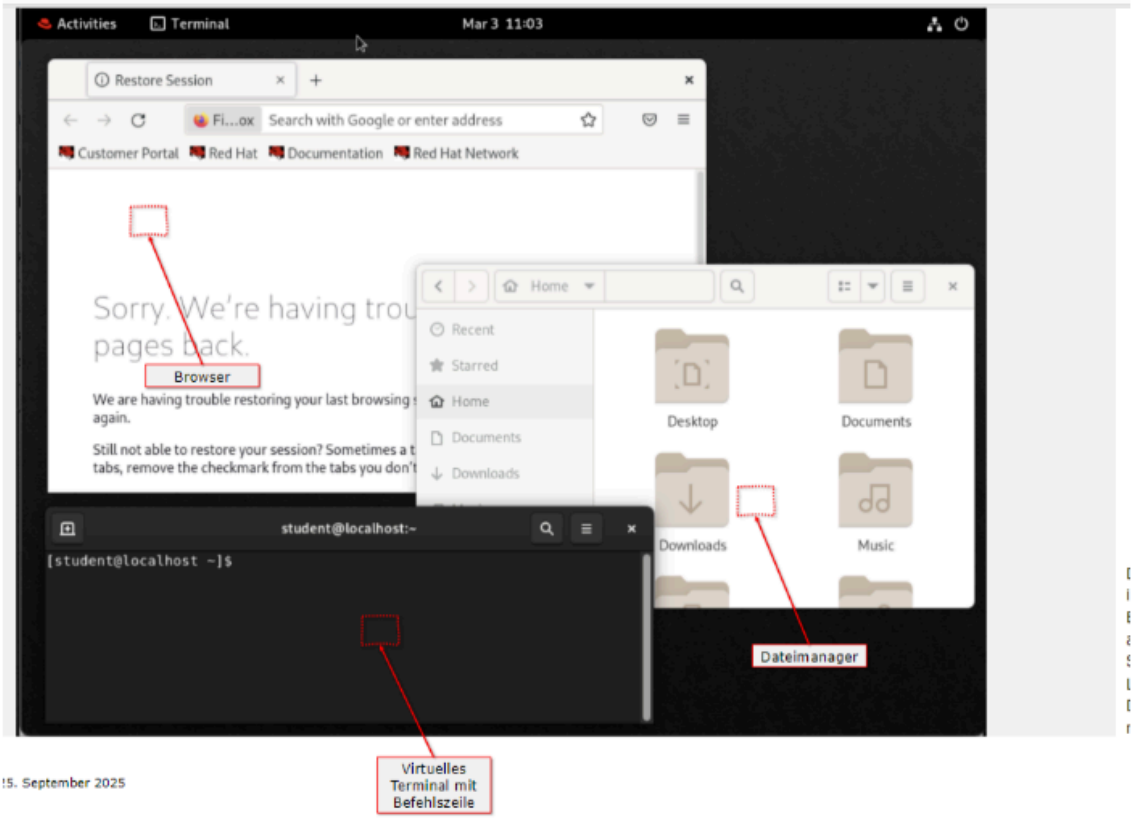
pty = pseudoterminal
- Kernfunktion: Eine Software-Schnittstelle, die ein Terminal simuliert.
- Zweck: Ermöglicht Kommunikation zwischen zwei Prozessen, wobei einer ein echtes Terminal „vortäuscht".
- Praxis: Wird genutzt, wenn kein direkter Hardware-Zugriff besteht, z. B. bei SSH-Sitzungen, xterm oder gnome-terminal.

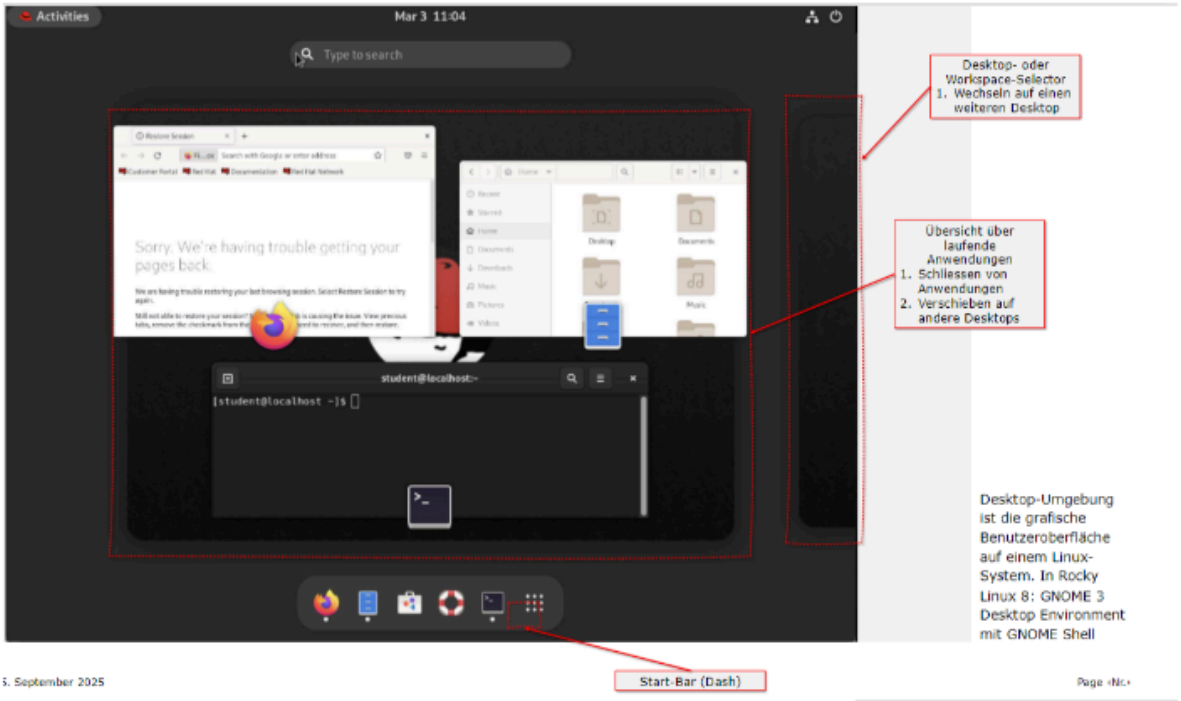
2.2. GNOME3-Bedienoberfläche

In Rocky Linux 8: GNOME 3 Desktop Environment mit GNOME Shell

GNOME Terminal ist ein Terminal-Emulator für die Befehlszeile und ein GUI.






Activities
Mar 3 11:04
Type to search
Sorry, We're having trouble getting your pages back.
student@localhost:~
[student@localhost ~]$
Desktop- oder Workspace-Selector
1. Wechseln auf einen weiteren Desktop
Übersicht über laufende Anwendungen
1. Schliessen von Anwendungen
2. Verschieben auf andere Desktops
Desktop-Umgebung ist die grafische Benutzeroberfläche auf einem Linux-System. In Rocky Linux 8: GNOME 3 Desktop Environment mit GNOME Shell
Start-Bar (Dash)
5. September 2025
Page ‹Nr.›

2.3. Befehlszeile mit Bash Shell

Standard Intepreter (Shell) in Rocky & Red Hat Linux ist **bash (GNU Bourne-Again Shell)**
Weitere Shells:

- zsh in MacOS
- Windows cmd.exe, PowerShell

Bash zum Ausführen von Befehlen sehr **leistungsfähig**

Bash stellt **Skriptsprache** bereit

Shell als **normaler Benutzer**:

```sh
[user@host ~]$
```

- `~` Home Verzeichnis

Shell als **Root / Superuser / Administrator**:

```sh
[user@host ~]#
```

2.3.1. Login im Textmodus

- Terminal nötig, um Shell zu starten
- Physical Terminal / Console
  - Textmodus
  - Mehrere virtuelle Konsolen ⟶ eigene Anmeldesitzungen
  - Wechsel: `Strg + Alt + F1–F6`
  - Rocky Linux 8.5:
    - `tty1 (F1)` : grafischer Login
    - `tty2–tty6 (F2–F6)` : Text-Terminals
  - Text-Konsolen: Login mit Benutzername + Passwort ⟶ Befehlszeile

2.3.2. Login über Netzwerk

- Linux-Administratoren: Shell-Zugriff oft über Netzwerk notwendig
- Moderne Umgebungen: viele Headless-Server, virtuelle Maschinen, Cloud-Instanzen
- Keine physischen Konsolen verfügbar
- Häufigste Methode: **Secure Shell (SSH)**
- Die meisten Linux-Systeme (inkl. RHEL) und macOS: OpenSSH-Programm `ssh` vorhanden

2.3.3. SSH-Authentifizierung

2.3.3.1. Passwort

- Anmeldung mit Benutzername + Passwort möglich
- Beispiel: `ssh remoteuser@remotehost`
- Passwort des aktuellen Benutzers wechseln / ändern: `passwd`

2.3.3.2. Schlüsselbasierte Authentifizierung

- Alternative zu Passwörtern (sicherer)
- Benutzer verwendet private Identitätsdatei (z. B. `key.pem`)
- Server enthält passenden öffentlichen Schlüssel
- Login ohne Passwort möglich
- Beispiel: `ssh -i key.pem remoteuser@remotehost`

2.3.3.3. Erste Verbindung zu neuem Host

- ssh meldet: **Authentizität des Hosts kann nicht festgestellt werden**
- Anzeige des **Key Fingerprint** (z. B. `ECDSA key fingerprint`)
- Benutzer muss entscheiden: `yes` oder `no`

2.3.4. Abmelden & Herunterfahren

2.3.4.1. Sitzung beenden

- `exit` ⟶ aktuelle Shell-Sitzung beenden
- `Strg+D` ⟶ Alternative zum Beenden

2.3.4.2. System herunterfahren (shutdown)

- Befehl: `shutdown [-h] [-r] Zeit [Nachricht]`
- Zeitangaben:
  - `hh:mm` ⟶ genaue Uhrzeit
  - `+mm` ⟶ Verzögerung in Minuten
  - `now` ⟶ sofort
- Optionen:
  - `-h` ⟶ halt (anhalten)
  - `-r` ⟶ reboot (neustarten)
- Beispiel: `sudo shutdown -h now` (Root-Rechte nötig)

2.3.5. Inhalte auflisten

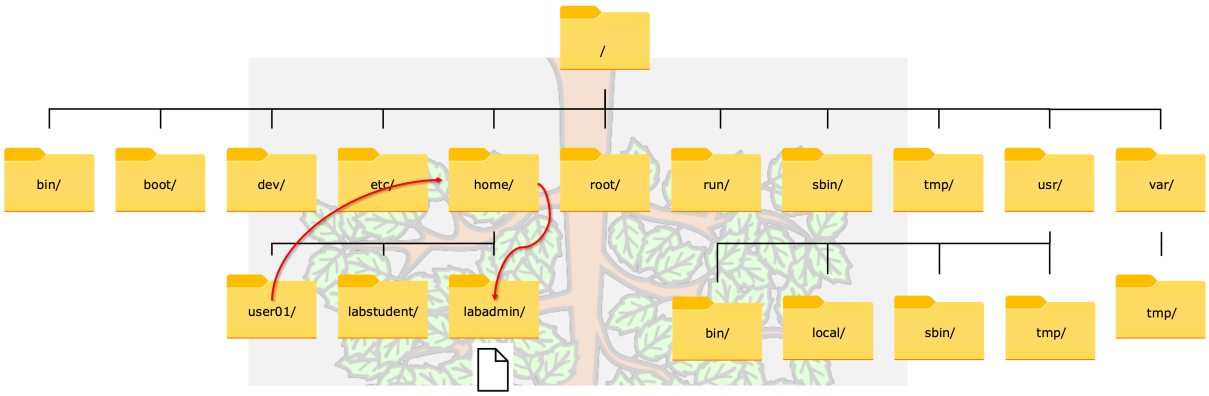
- `ls -l`: Listet den Inhalt eines Verzeichnisses im Langformat auf (Bsp. `ls -l /home`)
- `ls -lia`: Listet Inhalte mit Inode-Nummer (`-i`), im Langformat (`-l`) und inklusive versteckter Dateien (`-a`) auf (Bsp. `ls -lia /etc /home /var`)

3. Dateien und Verzeichnisse

3.1. Linux Dateisystem Aufbau

In Linux Dateisystem als umgekehrter Baum aufgebaut.

Wurzel ist das `/` Verzeichnis



absoluter Pfad: durchläuft **gesamten Baum bis zur Dateieben** (vom Stamm aus)
- `/home/labstudent/file`

relativer Pfad: durchläuft gesamten Baum **vom aktuellen Verzeichnis** aus
- `../labadmin/file`

Ein Verzeichnis hat mindestens zwei Referenzen:
- `.` : Referenz auf sich selbst
- `..` : Referenz auf übergeordnetes Verzeichnis

Wenn sich der **relative Pfad** auf ein Unterverzeichnis oder eine Datei **im aktuellen Verzeichnis** bezieht, **wird `./` oft weggelassen**

`./` ist nur wirklich erforderlich, um eine **ausführbare Datei im aktuellen Verzeichnis** zu starten

3.1.1. `/bin`

`/bin` Synonym zu `/usr/bin` (symbolischer Link)

Programme / Binaries

3.1.2. `/boot`

Dateien, die **zum Starten und Bootvorgang** nötig sind.
- Kernel, Bootloader

3.1.3. `/dev`

Enthält **Gerätedateien (device files)**

- Dient dem **Zugriff auf Hardware** (z. B. Maus, Tastplatte, Tastatur, etc.)
- Stellt eine **Schnittstelle** dar, die der Kernel bereitstellt
- Ermöglicht Kommunikation mit dem Kernel über **virtuelle Dateien**

3.1.4. `/etc`

Konfigurationsdateien und **Dateien mit Einstellungen** für das **gesamte System**
- „etc“ früher „et cetera“, heute „**editable text configuration**“

3.1.5. `/home`

Speicherung von **persönlichen Daten und Konfigurationsdateien** von **normalen Benutzer**

3.1.6. `/root`

Home-Verzeichnis für Superuser `root`

3.1.7. `/run`

Enthält **Laufzeitdaten** für Prozesse seit dem letzten Start
- z. B. **Prozess-ID-Dateien** und **Lock-Dateien** (Sperrdateien)
- **Wird beim Neustart neu erstellt** (Inhalt geht verloren)
- Konsolidiert die früheren Verzeichnisse `/var/run` und `/var/lock`

3.1.8. `/sbin`

Synonym zu `/usr/sbin` (symbolischer Link), **System Binares - Systemprogramme**

3.1.9. `/tmp`

Systemweit beschreibbarer Speicherplatz für **temporäre Dateien**
- Dateien werden **automatisch gelöscht**
- **10 Tage** lang nicht genutzt, geändert oder aufgerufen wurden → gelöscht

Zusätzliches temporäres Verzeichnis: `/var/tmp`
- Dateien werden gelöscht, wenn sie **länger als 30 Tage** nicht genutzt oder geändert wurden

3.1.10. `/usr`

Beinhaltet:
- **Installierte Software**
- **Gemeinsam genutzte Bibliotheken** (`/usr/lib`)
- **Include-Dateien**
- **Schreibgeschützte Programmdaten**

3.1.10.1. Wichtige Unterverzeichnisse

- `/usr/bin` → Benutzer-Commands für die Befehlszeile
- `/usr/sbin` → Systemverwaltungs-Commands für root (z. B. `shutdown`)
- `/usr/local` → Lokal angepasste Software
  - Pakete, die **nicht von der Distribution** kommen
  - Selbst kompilierte Programme in `/usr/local/bin`
- Weitere:
  - `/usr/man` → Handbuchseiten der Programme
  - `/usr/share` → Gemeinsame Daten (z. B. Bilder, Texte)

3.1.11. `/var`

variable Daten, die sich dynamisch ändern, Daten bleiben **zwischen Bootvorgängen** bestehen

Typische Inhalte:
- **Datenbanken**
- **Cache-Verzeichnisse**
- **Protokolldateien (Logs)**
- **Vom Drucker gespoolte Dokumente**
- **Website-Inhalte**

3.1.12. statischer Inhalt

bleibt unverändert, bis er explizit bearbeitet oder neu konfiguriert wird

- `/bin`
- `/boot`
- `/etc`
- `/home`
- `/root`
- `/sbin`
- `/usr`

3.1.13. dynamische oder variable Inhalte

können durch aktive Prozesse modifiziert oder angehängt werden

- `/dev`
- `/run`
- `/var`

3.1.14. Laufzeitinhalte

Prozess- oder systemspezifische Inhalte, werden **bei einem Neustart gelöscht**

- `/run`
- `/tmp`
- `/var`

3.2. Befehle für Dateisystem

Befehl	Beschreibung
`whoami`	Zeigt aktuellen Benutzer an
`date`	Gibt aktuelles Datum und Uhrzeit aus `Sat Jan 26 08:13:50 IST 2019`
`date +%R`	Uhrzeit `08:13`
`date +%x`	Datum im Kurzformat `01/26/2019`
`date +%F`	Datum im Format `JJJJ-MM-TT`
`date +%s`	Anzahl Sekunden seit der Epoche, 1970-01-00 00:00 UTC (Unix-Timestamp)
`pwd`	Aktuelles Arbeitsverzeichnis anzeigen
`ls`	Dateien/Verzeichnisse auflisten
`ls -l`	Detaillierte Ansicht (Langformat)
`ls -a`	versteckten Dateien
`ls -i`	Inode Nummer
`ls -lai`	Detailliert mit versteckten Dateien & Inode Nummer
`ls -R`	Rekursiv alle Unterverzeichnisse
`cd <Verz>`	Verzeichnis wechseln (einfacher Wechsel)
`cd /pfad/zum/verz`	Absoluter Pfadwechsel
`cd ..`	Ein Verzeichnis nach oben
`cd../..`	Arbeitsverzeichnis zwei Ebene nach oben vom aktuellen Ort ändern
`cd -`	Zum vorherigen Verzeichnis zurück
`cd` (ohne Argument)	Zurück ins Home-Verzeichnis
`cat <datei>`	Datei-Inhalt ausgeben
`head <datei>`	Erste Zeilen einer Datei anzeigen
`wc <datei>`	Zeilen, Wörter, Zeichen zählen
`wc -l <datei>`	Zeilenanzahl zählen
`file <datei>`	Dateityp ermitteln
`history`	Liste der zuletzt ausgeführten Befehle
`!<Zahl>`	Befehl aus History erneut ausführen
`!<string>`	Letzten passenden Befehl ausführen (z. B. `!file`)
;	Führt mehrere Befehle nacheinander aus, unabhängig vom Erfolg. (vor und nach dem Semikolon ein Leerschlag)
%%	Verknüpft Befehle, zweiter wird nur bei Erfolg des ersten ausgeführt
||	Wenn der erste Befehl fehlschlägt, wird dann der zweite ausgeführt
`!string`	letzer Befehl erneut ausführen, anhand Befehlsname
`!zahl`	bestimmten Befehl in der Verlaufsliste erneut ausführen

3.3. Umgang mit Dateien und Verzeichnissen

3.3.1. Allgemeinde Dateiverwaltungsbefehle

Befehl	Beschreibung
`mkdir <verzeichnis>`	Erzeugt ein Verzeichnis
`mkdir -p <Pfad>`	Erstellt auch verschachtelte Verzeichnisse, d.h. alle übergeordneten Verzeichnisse, die noch nicht existieren, werden automatisch mit erstellt.
`cp <Datei> <neue Datei>`	Datei kopieren
`cp -r <Verzeichnis> <neues Verzeichnis>`	Verzeichnis und Inhalt kopieren
`mv <Datei> <Datei>`	Verzeichnis oder Datei verschieben oder umbenennen
`rm <Datei>`	Datei löschen
`rm -r <Verzeichnis>`	Verzeichnis löschen mit Dateien
`rmdir <Verzeichnis>`	Leeres Verzeichnis löschen
`touch <Datei>`	Erstellt eine neue, leere Datei. Falls die Datei bereits existiert, wird das Zugriffs- und Änderungsdatum aktualisiert. -> mit geschweiften Klammern können mehrere Dateien hintereinander erstellt werden
`mkdir ~/<Pfad>/<name>_$(date +%F)`	Verzeichnis mit aktuellem Datum erstellen (`yyyy-mm-dd`)

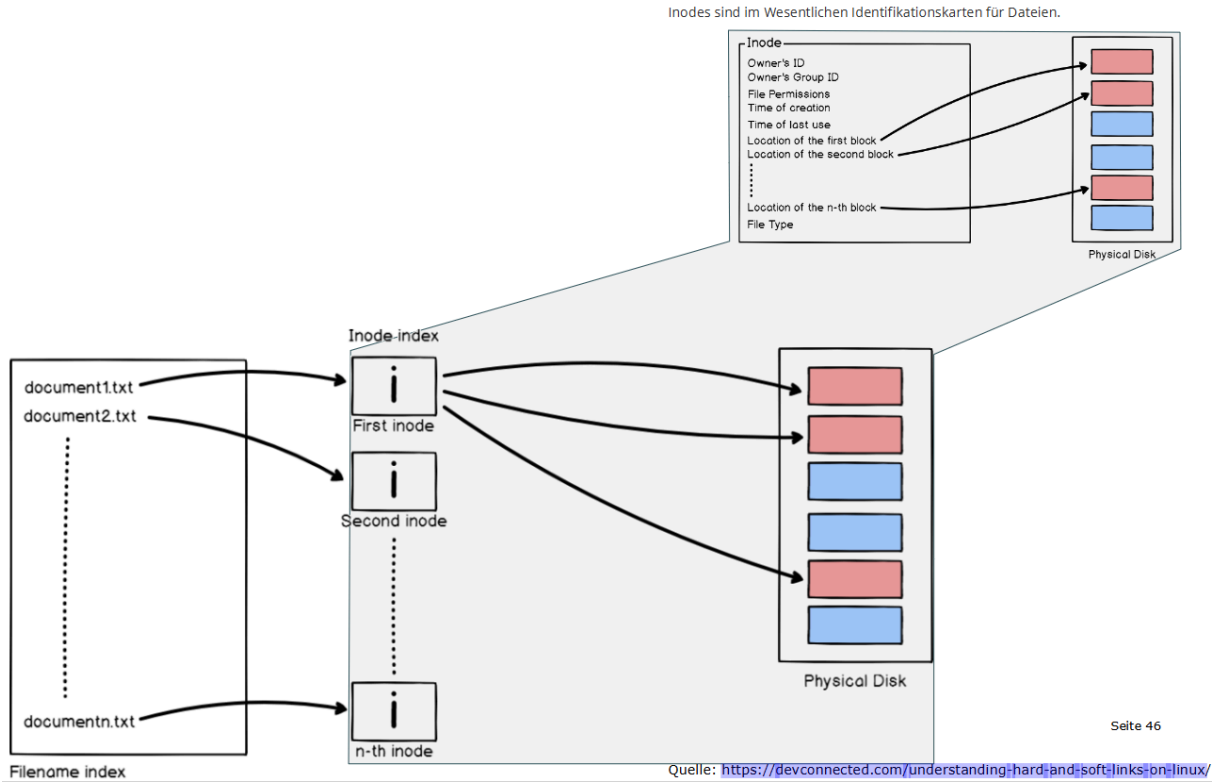
3.4. Soft und Hardlinks

Mit Soft und Hard Links können mehrere Namen für dieselbe Datei erstellt werden.

Grundlegender Aufbau vom Linux Dateisystem:

- Dateinamen werden vom Dateisystem in einem separten Index gespeichert
- Metainformationen (Berechtigungen, Besitzer, Zeitstempel) werden in Inodes gespeichert
- Inodes verweisen wiederum auf die physischen Datenblöcke verweisen, in denen die eigentlichen Daten liegen

Inodes = Identifikationskarten für Dateien



3.4.1. Soft Link (symbolischer Link):

Tatsächlicher Link auf die original Datei

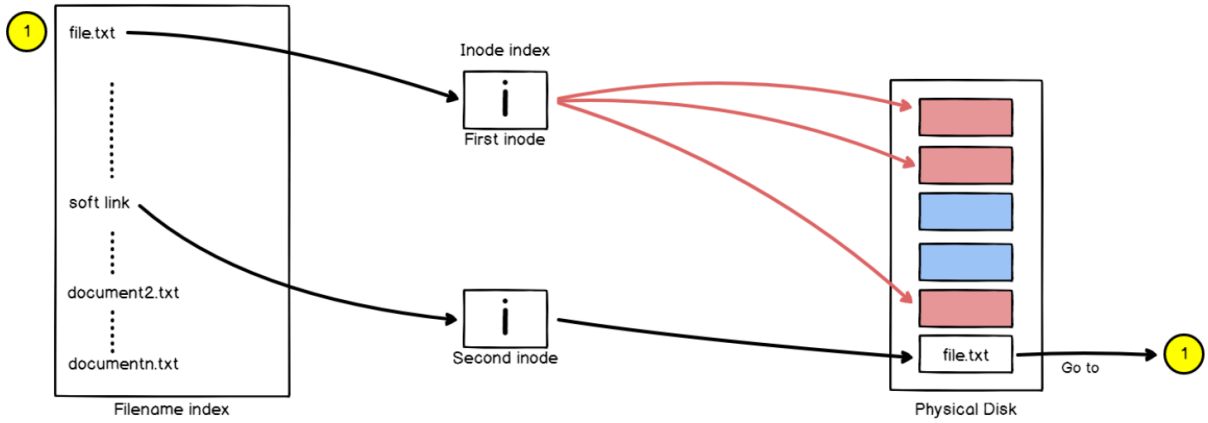
-> wird die Original Datei gelöscht funktioniert der Soft Link nicht mehr

- ähnlich wie eine Verknüpfung unter Windows

Ein Soft Link:

- kann das Dateisystem überschreiten
- ermöglich Verlinkungen zwischen Verzeichnissen
- hat eine andere Inode-Nummer und andere Berechtigungen als die Original Datei

In einer `ls -l` Auflistung erkennbar am gesetzten `l`-Bit am Anfang der Berechtigungen und dem Pfeil (`->`), der auf das Ziel zeigt.



Softlinks, auch symbolische Links genannt, sind Dateien, die auf andere Dateien im Dateisystem verweisen.

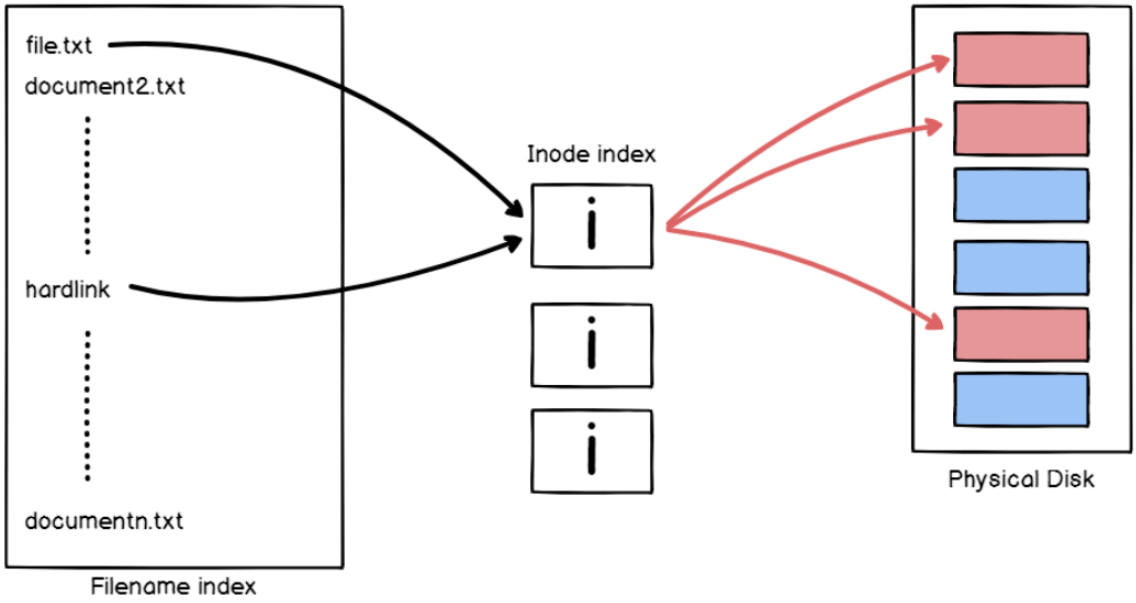
3.4.2. Hard Link:

Zweiter Verweis auf die Original Daten

-> wird die Original Datei gelöscht enthält der Hardlink immer noch die Original Daten

Ein Hard Link:

- kann die Dateisystemgrenzen nicht überschreiten
- kann keine Verzeichnisse verknüpfen
- hat dieselbe Inode-Nummer und Berechtigung wie die Original Datei
- Brechtigungen werden aktualisiert wenn die Berechtigung der Originaldatei angepasst werden



Hardlinks zu einer Datei sind Instanzen der Datei unter einem anderen Namen im Dateisystem.

3.4.3. Nützliche Befehle

Befehl	Beschreibung
`ln <Original Datei> <Neue Datei>`	Erzeugt einen Hardlink
`ln -s <Original Datei> <Neue Datei>`	Erzeugt einen Softlink
`ls -i -l`	Zeigt die Inode Informationen zu den Dateien an und wie viele Links auf die Datei existieren
`cat <Datei>`	Inhalt der Datei auf der Konsole ausgeben

```bash
-rw-r--r--. 2 labstudent labstudent 15 Jan 30 13:08 hard.txt
-rw-r--r--. 2 labstudent labstudent 15 Jan 30 13:08 Original.txt
lrwxrwxrwx. 1 labstudent labstudent 12 Jan 30 13:10 soft.txt -> Original.txt
```

2 Datei Links bei einem Hardlink bedeutet, dass es zwei Dateien gibt (1x den Hardlink und 1x die Originaldatei)

3.5. Tilde-Präfix `~`

- Platzhalter für das Homeverzeichnis, z.B. `/home/labstudent`
- mit `echo ~` ausgeben, für was tilde steht
- `~root` Home des root Benutzers verwenden
  - wenn nichts angeben wird, wird das Homeverzeichnis des aktuellen Benutzers genommen

3.6. Bash Shell Expansion

nicht mit regulären Ausdrücken verwechseln

Bash bietet verschiedenen Möglichkeiten einen Befehl zu erweitern. Dafür werden Platzhalterzeichen verwendet:

Muster	Wirkung
`*`	Eine Zeichenfolge mit null oder mehr Zeichen
`?`	Ein einzelnes Zeichen
`img{jpg,png}`	Erzeugt `img.jpg img.png`
`test{1..4}`	Erzeugt `test1 test2 test3 test4`
`[abc..]`	Jedes Zeichen in der eckigen Klammer
`[!abc..]`	Jedes Zeichen welches sich **nicht** in der eckigen Klammer befindet
`[^abc..]`	Jedes Zeichen welches sich **nicht** in der eckigen Klammer befindet
`[[:alpha:]]`	Alle Zeichen des Alphabets
`[[:lower:]]`	Alle Zeichen in Kleinbuchstaben
`[[:upper:]]`	Alle Zeichen in Grossbuchstaben
`[[:alnum:]]`	Alle Zeichen des Alphabets oder Ziffern
`[[:punct:]]`	Alle druckfähigen Zeichen (ohne Leerzeichen und alphanumerische Zeichen)
`[[:digit:]]`	Eine Ziffer zwischen 0 und 9
`[[:punct:]]`	Alle Leerräume
`$(date)`	Führt einen Befehl aus und fügt dessen Ausgabe in die Zeile ein. z.B. hier `date`

3.6.1. Beispiele

zwei Fotos erstellen

```sh
touch {baden-1,baden-2}_seebad_luzern.jpg
```

Fortlaufende Dateien erstellen mit Nummerierung

```sh
touch urlaubsvideo_season{1..2}_episode{1..6}.ogg
```

Dateien verschieben

```sh
mv urlaubsvideo_season*.ogg ../Sommer_Urlaubs-Videos/
```

3.7. Hilfetools und Dokumentationsseiten

Für alle installierten Befehle gibt es Handbuchseiten: **man page**

- man pages werden zusammen mit Softwarepaketen installiert
- sie können mit dem `man` Befehl aufgerufen werden
- man pages sind nur geringfügig standardisiert
- man pages werden in **Sections** eingeteilt
- man pages liegen unter `/usr/share/man`

3.7.1. Hilfebefehle

`apropos` -> Stichwortsuche

alternativ: `man -k`

Optionen:

Option	Beschreibung
`-s, --sections`	Durchsucht nur eine bestimmte Section z.B: `apropos -s 5,9 pass` 5 & 9 nach „pass“ suchen
`-a, --and`	Zeit nur das Element, das mit allen angegebenen Schlüsselwörter übereinstimmt

Um die Whatis Datenbank neu zu laden: `sudo mandb -c`

`whatis` -> Gibt eines Kurzfassung der Man Page aus

-> whatis benötigt vollständige/genaue Parameter

3.7.2. Sections

Es gibt folgende 8 Sections:

1. Ausführbare Programme oder Befehle
2. Systemaufrufe (Kernel)
3. Bibliotheksaufrufen
4. Spezielle Dateien (/dev)
5. Dateiformate und Konventionen (Konfigurationsdateien)
6. Spiele
7. Sonstiges (z.B. `man` selbst)
8. Systemverwaltungsbefehle (nur für root)
9. Kernel-Routinen

3.7.3. Navigation

Mit folgenden Tastenkombinationen kann eine `man` Page navigiert werden

Befehl	Navigation
Leertaste	Ein Bildschirm vorwärts
pg dn	Ein Bildschirm vorwärts
pg up	Ein Bildschirm zurück
↓	Eine Zeile vorwärts
↑	Eine Zeile rückwärts
d	einen halben Bildschirm vorwärts scrollen
u	einen halben Bildschirm rückwärts scrollen
/Zeichenfolge	Suche
n	Vorwärts suchen
N	Rückwärts suchen
g	Anfang der man Page
G	Ende der Man Page
q	Beenden

3.7.4. GNU info

info ist ein in C umgesetztes Dokumentationssystem von GNU basierten Betriebssystemen

-> info Dokumente enthalten Erklärungen und Beispiele

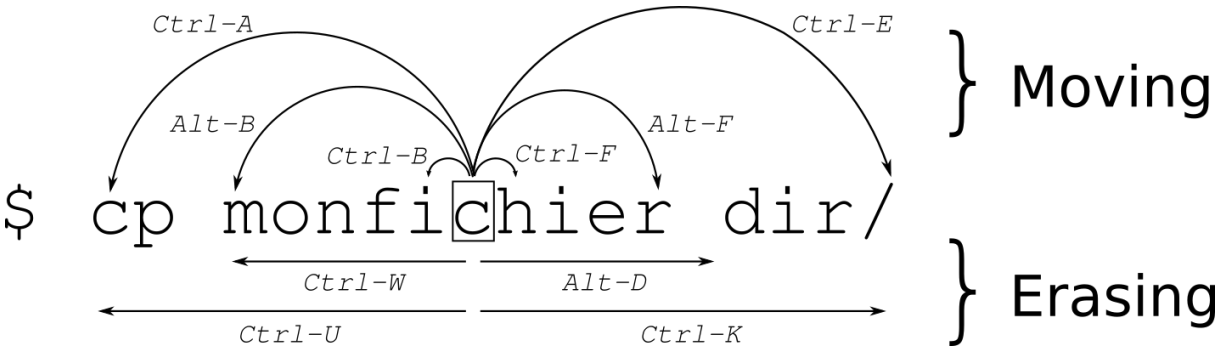
Ein Viewer wäre `pinfo`

Zweit weitere Hilfetools: `cheat` und `tldr`

3.8. Bash Shortcuts

Tab Completion: vervollständigt Befehle oder Dateinamen -> zwei Mal drücken: Vorschläge

Tastenkombination	Beschreibungen
`CTRL + A`	An den Anfang der Zeile springen
`CTRL + E`	Ans Ende der Zeile springen
`CTRL + ←`	Ein Wort nach links springen
`CTRL + →`	Ein Wort nach rechts springen
`CTRL + U`	Alle Zeichen links vom Cursor löschen / Komplette Zeile löschen
`CTRL + K`	Alle Zeichen rechts vom Cursor löschen
`CTRL + H`, Rücktaste	ein Zeichen nach links löschen
`CTRL + D`, delete	ein Zeichen nach rechts löschen
`CTRL + W`	Wort links vom Cursor löschen
`CTRL + R`	History durchsuchen
`CTRL + D`	Exit
`ESC + .`	Letztes Wort des vorherigen Befehls
`CTRL + L`	Bildschirminhalt im Terminal löschen
`!!`	letzter Befehl wiederholen



4. Lesen und Bearbeiten von Dateien aus der Befehlszeile

4.1. Programmoutputumleitung (Pipes)

4.1.1. Eingabe / Ausgabe-Kanäle (Dateideskriptoren)

Ausgabe: Zeigt die Ausgabe auf dem Bildschirm an

Eingabe: Prozess/Befehl nimmt Eingabe vom Benutzer über die Tastatur entgegen

- Linux verwendet Dateideskriptoren (nummerierte Kanäle) um Eingabe zu erhalten und Ausgaben zu senden
- Alle Prozesse beginnen mit mind. drei Dateideskriptoren
  - Kanal 0 `stdin` (Standardeingabe): liest Eingabe von der Tastatur
  - Kanal 1 `stdout` (Standardausgabe): sendet eine normale Ausgabe an das Terminal
  - Kanal 2 `stderr` (Standardfehler): sendet Fehlermeldungen an das Terminal
  - möglicherweise höhere Kanäle für separate Verbindungen zu anderen Dateien

Programm(output)umleitung -> Mit Hilfe von Dateideskriptoren kann man die Eingabe oder Ausgabe auf andere Kanäle umleiten

Beispiele:
- Output-Umleitung nach `/dev/null` -> wird ignoriert
- Output-Umleitung in Datei -> wird gespeichert

Nummer	Name	Beschreibung
0	`stdin`	Eingabe zum Prozess (z.B. Tastatur)
1	`stdout`	Normale Ausgabe (z.B. Informationen auf die Konsole)
2	`stderr`	Ausgabe von Fehlern (z.B. auf die Konsole)
3, 4, 5, ...	`<filename>`	Andere Dateien zum Lesen oder Schreiben

4.1.1.1. Beispiel

`1>` / `>`: um Standardausgabe des Befehls umzuleiten -> wenn keine Zahl angegeben wird, wird von 1 ausgegangen

```bash
[labstudent@lios-01 ~]$ ls 1> stdout.txt

[labstudent@lios-01 ~]$ ls > stdout.txt
```

Der Befehl erstellt eine Datei und speichert die Ausgabe von `ls` in der Datei `stdout.txt`. Keine Ausgabe auf dem Bildschirm.

`2>`: um Fehler umzuleiten

```bash
[labstudent@lios-01 ~]$ cat myfile.txt 2> stderr.text
```

`1>  2>`: Umleitung der Standardausgabe und des Standardfehler

```bash
[labstudent@lios-01 ~]$ ls 1> stdout.txt 2> stderr.txt
```

Verwendung	Beschreibung
`> file`	Umleitung von **stdout** (1) in Datei; Datei wird **überschrieben**
`1> file`	Umleitung von **stdout** (1) in Datei; Datei wird **überschrieben**
`>> file`	Umleitung von **stdout** (1) in Datei; **Anhängen** am Ende der Datei
`2> file`	Umleitung von **stderr** (2) in Datei; Datei wird **überschrieben**
`2> /dev/null`	**Verwerfen** von Fehlermeldungen durch Umleitung nach `/dev/null`
`>/dev/null 2>&1`	alle Ausgaben und Fehlermeldungen verwerfen
`1> file1 2> file2`	**stdout** (1) wird in `file1` geschrieben, **stderr** (2) in `file2`
`> file 2>&1`	Umleitung von **stdout** (1) und **stderr** (2) in Datei; Datei wird **überschrieben**
`>> file 2>&1`	Umleitung von **stdout** (1) und **stderr** (2) in Datei; **Anhängen** am Ende der Datei
`2>&1`	stderr (2) wird nach stdout (1) geleitet, das & ist wichtig, sonst würde 1 als Datei und nicht stdout interpretiert

ACHTUNG: Reihenfolge spielt eine Rolle, von links nach rechts

```bash
utility 2>&1 >output.log
```

- Standardfehlerstream `stderr` wird zuerst dorthin umgeleitet, wo der Standardausgabestream gerade jetzt `stdout` hingeht (möglicherweise Konsole)
- dann wird der Standardausgabestream `stdout` in eine Datei umgeleitet
- Standardfehlerstream wird **nicht** zu dieser Datei umgeleitet

```bash
utility <output.log 2>&1
```

- Standardausgabestream `stdout` wird in `output.log` geschrieben
- Standardfehlerstream `stderr` wird nach `stdout` umgeleitet, was `output.log` ist
- Standardfehlerstrom wird mit dem Standardausgabestream in die Datei umgeleitet

```bash
utility 1>&2 >output.log
```

- Ausgabe wird dort hin geleitet wo Fehler gerad jetzt hingeleitet wird (Konsole)
- Ausgabe wird in output.log geschrieben (erste Anweisung wird überschrieben)

Korrekt wäre:

```bash
utility >output.log 2>&1
```

oder

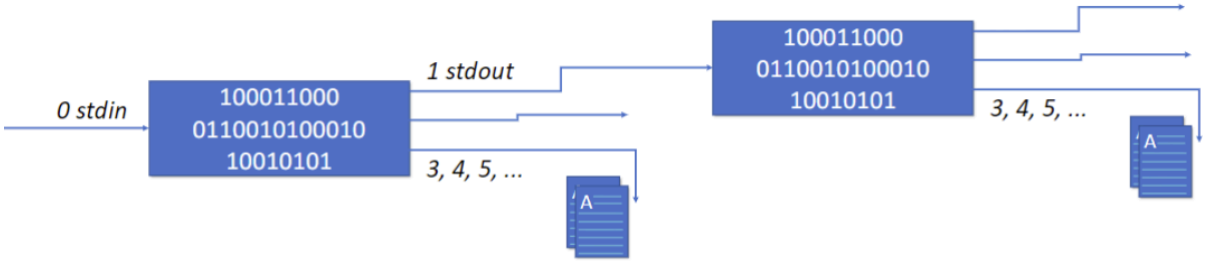
```bash
utility &>output.log
```

4.1.2. Linux-Pipes

Pipelines (Pipes) und Umleitungen manipulieren die Standardausgabe und die Standardeingabe.

- Umleitung sendet die Ausgabe an Dateien oder ruft die Standardeingabe von Dateien ab
- Pipes senden die Standardausgabe eines Prozesses an die Standardeingabe eines anderen Prozesses

```bash
[labstudent@lios-01 ~]$ cat /etc/passwd | wc -l
```



4.1.2.1. Beispiel

```bash
cat /var/log/syslog |          # Liest die System-Logdatei
grep -i "error" |              # Sucht nach Zeilen mit dem Wort "error"
awk '{print $5}' |             # Extrahiert das fünfte Wort (Dienstnamen)
sort |                         # Sortiert die Dienstnamen alphabetisch
uniq -c |                      # Zählt die Vorkommen
sort -nr |                     # Sortiert, zeigt die häufigsten Fehler zuerst
head -n 10                     # Zeigt die obersten 10 häufigsten Fehler an
```

Einzeiler:

```bash
sudo cat /var/log/syslog | grep -i "error" | awk '{print $5}' | sort | uniq -c | sort -nr
| head -n 10
```

4.2. Einführung in VIM (Vi Improved)

4.2.1. Geschichte

Bill Joy

- Mitgründer von Sun Microsystems
- Entwickler des Texteditors vi (1976)
- Wichtiger Beitrag zu BSD Unix
- Visionär für Open-Source und Netzwerktechnologien

Bram Moolenaar

- Hauptentwickler von Vim (1991)
- Vim: Verbesserung von vi, erweitert um viele Funktionen
- Starkes Engagement für Free Software
- Führte Vim zu einer der populärsten Texteditoren weltweit.

4.2.2. VIM (Vi Improved)

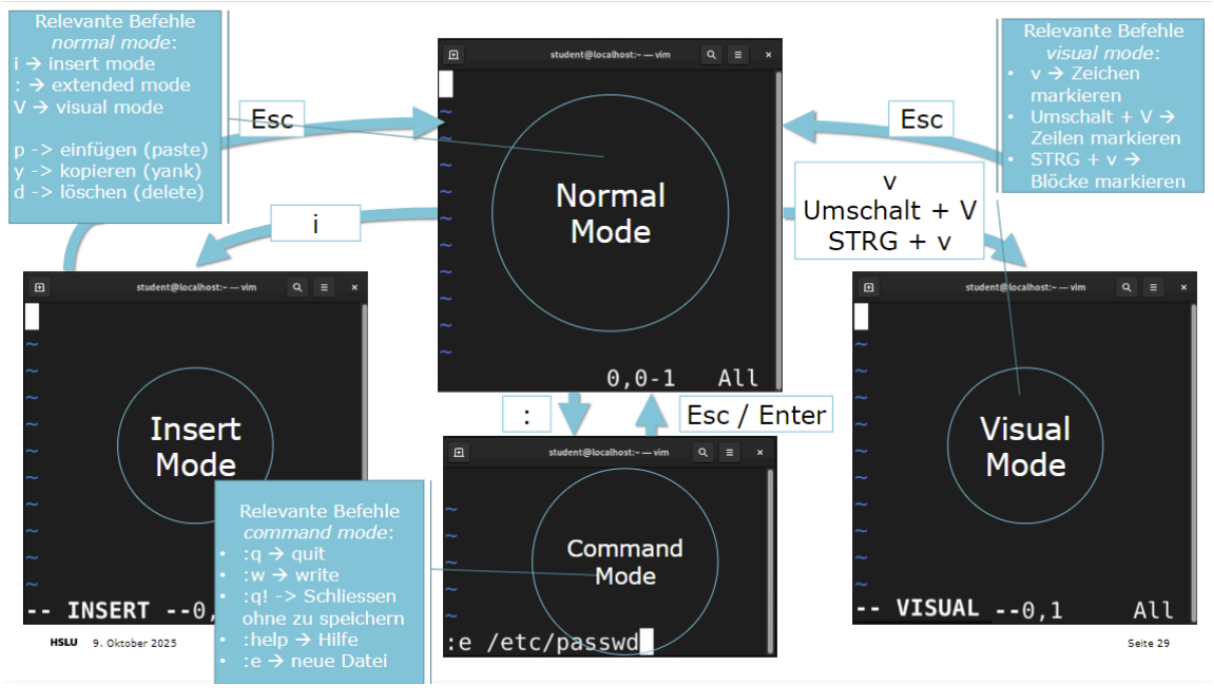
- Vim (Vi IMproved) ist eine Weiterentwicklung des Texteditors vi.
- 1991 von Bram Moolenaar veröffentlicht
- Verschiedene **Betriebs-Modi**
  - andere Texteditoren kennen nur einen kombinierten Modus für Eingabe und Befehle
- **Notwendigkeit**: viele Konfigurationsdateien liegen in Linux als Text-Datei vor (plain text, YAML, XML)
- Warum VIM: mind. einen Texteditor muss man kennen
- **Vorteile**:
  - GUI oder CLI-Modus
  - POSIX-Standard kompatible
  - Verfügbar auf vielen anderen Betriebssysteme wie MacOS

4.2.3. Installation auf Rocky Linux

- zwei verschiedene Arten für die Installation
  - Auswirkungen auf Funktionen und verfügbare Befehle
- auf unserem Server ist nur `vim-minimal` - Paket installiert
  - sehr einfache Installation
- Dateien mit Dateiname `vi` öffnen

4.2.4. Befehle

Befehl	Beschrieb
`vim <Datei>`	Vim öffnen



4.2.4.1. 1. Normal Mode (Relevante Befehle)

Befehl	Beschrieb
`i`	Wechsel zu Insert Mode, Einfüge-Modus
`:`	Wechsel zu Extended Mode, Erweiterter-Modus
`V`	Wechsel zu Visual Mode, Visueller Modus
`p`	paste, einfügen
`y`	yank, kopieren
`d`	delete, löschen

4.2.4.2. 2. Visual Mode (Relevante Befehle)

Befehl)	Beschrieb
`v`	zeichen markieren
`Umschalt + V`	Zeilen markieren
`STRG + v`	Blöcke markieren

4.2.4.3. 3. Command Mode (Relevante Befehle)

Befehl	Beschrieb
`:q`	quit, beenden
`:w`	write, speichern
`:q!`	schliessen ohne zu speichern
`:help`	Hilfe
`:e`	edit, neue Datei

4.2.4.4. Moduswechsel (Schlüssel)

von Modus	nach Modus	Taste
Insert / Visual Mode	Normal Mode	`Esc`
Command Mode	Normal Mode	`Esc` / `Enter`
Normal Mode	Insert Mode	`i`
Normal Mode	Visual Mode	`v`

4.2.5. Tastenkombinationen

4.2.5.1. Normal mode

Tastenkombination	Beschrieb
`i`	in Insert Mode wechseln
`:`	in Extended Command Mode wechseln
`v`	in Visual Mode wechseln (Buchstaben markieren)
`Umschalt + V`	In Visual Mode wechseln (Zeilen markieren)
`STRG + v`	In Visual Mode wechseln (Block markieren)
`p`	paste, einfügen
`y`	yank, kopieren
`d`	delete, löschen
`/<Suchbegriff>`	Suchen nach Suchbegriff

4.2.5.2. Command mode

Tastenkombination	Beschrieb
`:q`	quit, beenden
`:w`	write, speichern
`:q!`	beenden ohne zu Speichern
`:help`	hilfe
`:e <file>`	neue Datei öffnen

4.2.5.3. Weiteres

Tastenkombination	Beschrieb
`:colorscheme <tab>`	Farbschema wechseln
`:tabedit <tab>`	Reiter/Tabs öffnen
`STRG + n`	Automatische Vervollständigung
`:Vexplore`	Dateibrowser öffnen
`:set spell` / `:set spelllang=en`	Rechtschreibprüfung aktivieren (Sprache: Englisch)

4.2.6. Makros in VIM

In Vim kann die Aufnahme von Tastatureingaben genutzt werden, um repetitive Aufgaben effizienter zu erledigen.

1. **Aufnahme wird gestartet**: Es wird `q` gefolgt von einem Buchstaben gedrückt (z.B. `q a`), um die Aufnahme in das Register a zu beginnen. Der Buchstabe dient als Speicherort für die Aufnahme.
2. **Aktionen werden durchgeführt**: Die Aktionen, die aufgenommen werden sollen, werden ausgeführt. Dies können Bewegungen, Änderungen oder andere Befehle sein.
3. **Aufnahme beenden**: `q`
4. **Macro abspielen**: `@` gefolgt von dem Buchstaben (z.B. `@a`)
5. **Wiederholtes Abspielen**: Zahl `@` Buchstabe (z.B. `2@a`)

4.2.7. neovim

- leistungsstarker Texteditor
- Weiterentwicklung von vim
- deckt moderne Benutzeranforderungen ab
- Erweiterte Unterstützung für plugins
- eingebettete Lua-Programmierschnittstelle

5. Fernverwaltung, Nutzerverwaltung und Rechtemanagement

5.1. Einführung zu OpenSSH

DEFINITION: Secure Shell oder SSH bezeichnet ein kryptographisches Netzwerkprotokoll für den sicheren Betrieb von Netzwerkdiensten über ungesicherte Netzwerke.

Nutzen:
- sichere Verbindung zu einem Remote-System herstellen (mit dem Befehl `ssh`)
- sich als bestimmter Benutzer authentifizieren
- mit diesem Benutzer eine interaktive Shell-Sitzung auf dem Remote-System abrufen
- mit dem Befehl `ssh` kann auch einen einzelnen Befehl auf dem Remote-System ausführen, ohne eine interaktive Shell auszuführen

Anwendungen
- Fernwartung, um lokal eine entfernte Kommandozeile verfügbar zu machen
- Ersatz für unsichere Methode Telnet
- Vertrauliche Übertragung von Daten und Dateien
- X11 kann über SSH transportier und somit gesichert werden

5.2. Umgang mit OpenSSH

Login auf `workstation` mit Passwort `Hslu123`

```bash
[labstudent@lios-01 ~]$ ssh -Y lios-workstation-01 -l labstudent
# oder
[labstudent@lios-01 ~]$ ssh -Y labstudent@lios-workstation-01
labstudent@lios-workstation-dev-01's password: Hslu123
```

- `Y` aktiviert X11-Forwarding (grafische Fenster über SSH).

eine Anwendung auf dem Remote-Host starten:

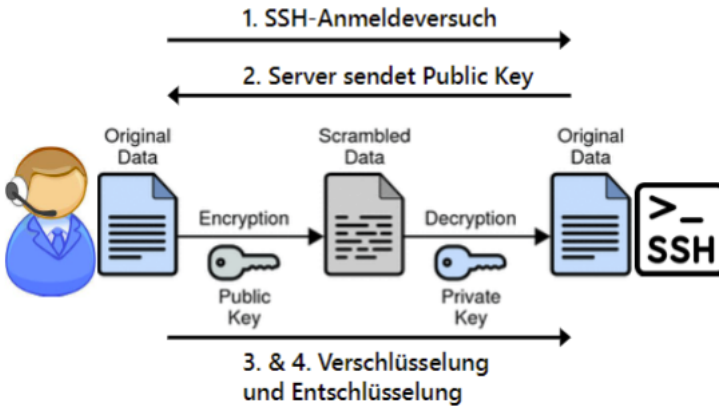
```bash
[labstuden@lios-workstation-01 ~]$ xeyes
```

Identifizieren vom Remote-Benutzern mit `w`:

```bash
[labstuden@lios-workstation-01 ~]$ w
```

5.3. SSH Public Key Subsystem

TL;DR: Kommunikation durch Verschlüsselung mit Public Keys (öffentliche Schlüssel)



1. Anmeldungsversuch
2. Server sendet Public Key
3. Client verschlüsselt mit Servers Public Key
4. Server kann Daten mit Privat Key entschlüsseln

5.3.1. Wichtige Dateien

5.3.1.1. `known_hosts` - Dateien

DEFINITION: Speicherort für die Identität von Remote-Servern auf den Clients

Zweck: Überprüfung der Identität von Remote-Servern

Ablauf: Client überprüft, ob eine Kopie des keys für diesen Server in den lokalen Dateien vorhanden ist
- `~/.ssh/known_hosts` : Benutzerverzeichnis des User
- `/etc/ssh/ssh_known_hosts` : Systemweit

Wenn keine Kopie vorhanden ist, fragt der Befehl `ssh` ob man sich trotzdem anmelden möchte, wenn ja, wird eine Kopie des Keys in `~/.ssh/known_hosts` gespeichert.

Sicherheit: Dient als Schutz vor Man-in-the-Middle angriffe

`StrictHostKeyChecking`
- Empfehlung: Setze den Wert auf `yes` , um Man-in-the-Middle-Angriffe zu verhindern.
- Dateien: `~/.ssh/config` (User) oder `/etc/ssh/ssh_config` (System)
- Effekt: Die Verbindung bricht sofort ab, falls der Public Key des Servers nicht übereinstimmt oder unbekannt ist.
- Vorteil: Verhindert das versehentliche Akzeptieren gefälschter Host-Keys.

5.3.1.2. `key.pub`

DEFINITION: Server speichert seine Public keys in dieser Datei unter `/etc/ssh/*.key.pub`

5.3.1.3. Public key ändert sich

zum Beispiel aufgrund eines Festplattenfehlers oder aus anderen legitimen Gründen -> die `know-hosts` -Dateien müssen manuell angepasst werden

5.3.2. Schlüsselbasierte Athentifizierung

SSH unterstützt passwort- und schlüsselbasierte Authentifizierung

Bei der schlüsselbasierten Authentifizierung:
- entfällt der interaktive Dialog zur Kennwort eingabe
- User erzeugt Signatur mit privatem Schlüssel, Server verifiziert mit öffentlichen Schlüssel
- Ermöglich Anmeldung per Skript

Befehle:
1. `ssh-keygen` SSH Schlüsselpaar generieren
2. `ssh-copy-id` exportieren des öffentlichen Schlüssels
3. `ssh-agent` hilft bei der Verwaltung der Schlüssel
4. `PasswordAuthentication no` in `/etc/ssh/sshd_config` wird die Passwort Authentifizierung deaktiviert

5.3.3. SSH Server Konfiguration

Der sshd Dienst implementiert das SSH Protokoll

Konfiguration in: `/etc/ssh/sshd_config` Nach eine Konfigurationsänderung:
`sudo systemctl restart sshd`

5.3.4. Best Practices

- kein Root Login erlauben `PermitRootLogin no`
- Schlüsselbasierte Authentifizierung verwenden
  - `ssh-keygen`
  - `ssh-copy-id`
- Kennwortbasierte Authentifizierung verbieten `PasswordAuthentication no`

6. Nutzerverwaltung

Linux ist ein Mehrbenutzersystem

Benutzer haben:
- einen einfache Usernamen
- eine interne ID (UID)
- normalerweise ein Passwort

Benutzer können:
- Dateieigentümmer sein
- Programme mit bestimmten Rechte ausführen

6.1. Nutzertypen

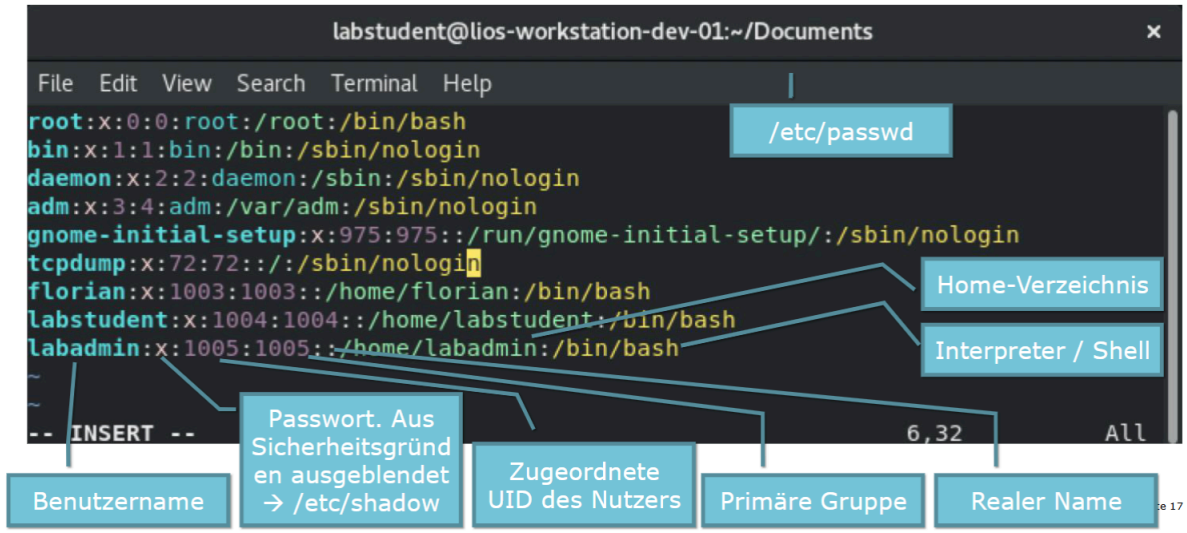
Root:
- UID = 0
- verwaltet das System
- vollständigen Zugriff auf das System

Systembenutzer:
- UID=1…200 (fix vergeben; bei jeder Installation gleich)
- UID=201…999 (dynamische erstellt beider Installation)
- führen unterstützende Systemprozesse aus
- kein Root
- spezifische Rechtezuordnung

Regulärer Benutzer
- UID 1000 bis 60000
- Alltägliche Arbeit
- eingeschränkter Zugriff

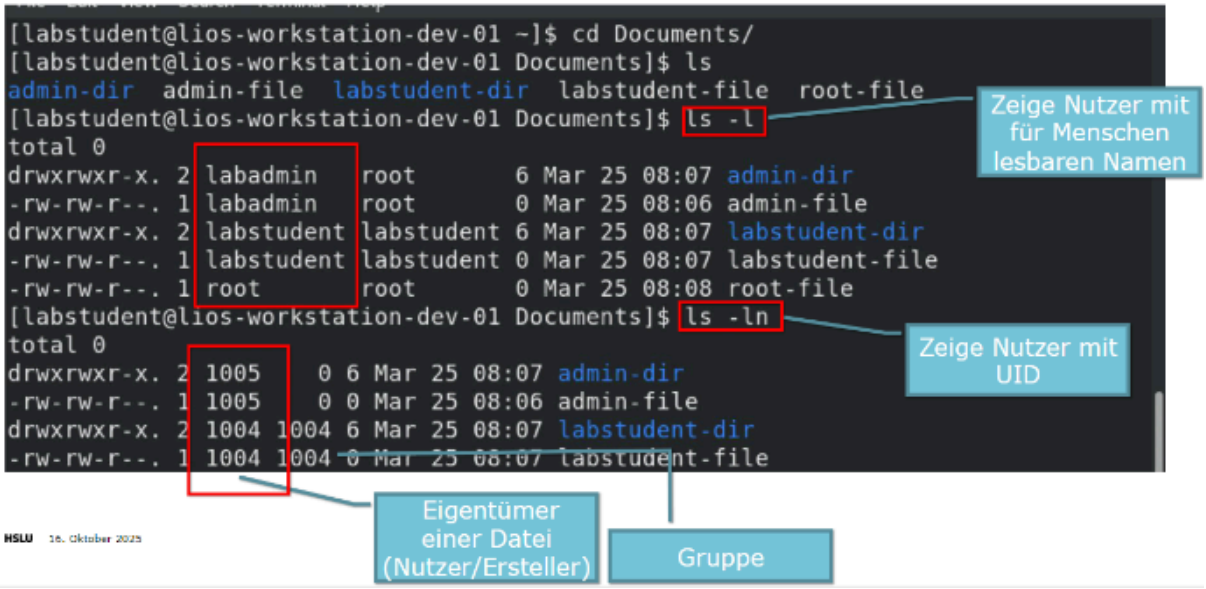
6.2. Befehle

6.2.1. `/etc/passwd`



`ls -l` -> Dateieigentümer mit Nutzernamen anzeigen

`ls -ln` -> Dateiegentümer mit UID



6.3. Gruppen

Jeder Nutzer gehört zu einer primären Gruppe

- einsehbar in `/etc/passwd`
- wird beim Erstellen eines Nutzers erstellt
- heisst oft wie der Nutzer selbst

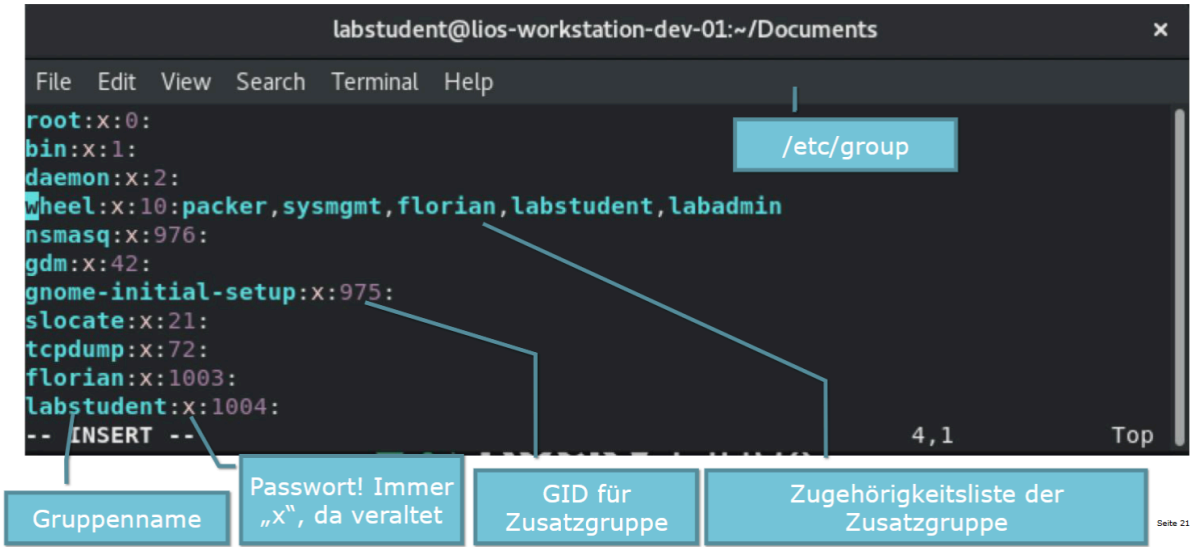
Zusätzlich können Nutzer Zusatzgruppen haben

- einsehbar in `/etc/group`

-> Zugriffsrechte für primäre oder Zusatzgruppen werden gleich behandelt

NutzerId und GroupID sind für die primäre Gruppe identisch

`/etc/group`



6.4. Superuser root

In Linux meldet man sich **nicht** mit root an

-> Es werden andere Mechanismen benutzt um in den Administratorenmodus zu wechseln

Der root User kann das System schwer beschädigen, da er allumfassende Rechte besitzt.

6.4.1. su, sudo, sudo -i, etc.

su (substitute user):

- startet eine andere Shell mit den Rechten des Zielbenutzers
- Abfrage des Passworts des Zielbenutzers

sudo (substitute user, do)

- basiert auf eine Konfigurationsdatei (`/etc/sudoers`) die auflistet welche Benutzer Rechte für bestimmte Aktionen haben
- Passwort vom Benutzer der die Shell gestartet hat
- Zum entziehen der Berechtigungen muss der Nutzer aus der Konfigurationsdatei entfernt werden
- **sauberere Verwaltung von Berechtigungen**
- Angabe zusätzliche Optionen möglich

-> Häufig hat der root Benutzer in RHEL Systemen gar kein Passwort da es nicht möglich ist sich als Root anzumelden

Befehl	Beschreibung
`su -` , `su -l`	neue Shell als Root starten (Neues Arbeitsverzeichnis, Neue Umgebungsvariablen)
`su -l labadmin`	Startet einen neues Shell als Benutzer labadmin
`sudo -i`	Startet eine neue Shell (ähnlich zu `su -l`)
`sudo -s`	Startet einen Shell im Kontext des aktuellen Nutzers
`sudo -u labadmin -i`	Startet einen neue Shell als labadmin
`sudo su`	Sudo Passwort wird abgefragt und verhindert dass das Passwort des Zielbenutzers abgefragt wird. **Besser `sudo -i` verwenden**

6.5. Hinzufügen, ändern und löschen von Benutzern

`useradd username` -> Erstellt Benutzer (Standardwerte aus /etc/login.defs)

Optionen:

kurz	lang	Beschreibung
`-m`	–create-home	Erstellt Home-Verzeichnis
`-g`	–gid	Legt primäre Gruppe fest
`-u`	–uid	Legt UID fest
`-s`	–shell	Legt die Standard Shell fest
`-c`	–comment	Notizen
`-r`	system user	erstellt Systembenuzter

`usermod username` -> Ändert Einstellungen eines Nutzers

Optionen:

kurz	lang	Beschreibung
`-d`	–home	Ändert Benutzerverzeichnis
`-g`	–gid	Legt primäre Gruppe fest
`-G <liste>`		Setzt die Zusatzgruppen
`-a`	–append	Fügt in Verbindung -G Gruppen hinzu anstatt sie zu ersetzen
`-c`	–comment	Notizen

`userdel -r username` -> löscht einen Nutzer, inklusive Nutzerverzeichnisse

6.5.1. Befehle aus Übungen

Befehl	Beschreibung
`wc -l /etc/passwd`	zählt Zeilen in einer Datei oder Ausgabe, hier wie viele Nutzer im System registriert sind
`getent passwd {1..999}`	Listet Systembenutzer auf
`getent passwd {1..999} | wc -l`	Anzahl Systembenutzer auflisten
`sudo useradd -m -d /home/student`	user mit dem Homeverzeichnis student hinzufügen
`tail -n 6 /etc/passwd`	letzten 6 Einträge anzeigen (die zuletzt erstellten)
`head -n 1 /etc/passwd`	erste Zeile ausgeben
`sudo -i - u student`, dann `id`	als student anmelden und neue shell starten, ID von student ausgeben

6.6. Datei `/etc/shadow`

Speichert verschlüsselte Passwort-Daten (als Hash-Codes) und Passwortinformationen.

```
benutzer:gehashtes passwort:doc:minimum:maximum: passwortvorwarndauer:deaktiv:deaktiv
seit
```

- **benutzer → der Username**
- **gehashtes passwort → das verschlüsselte Passwort des Benutzers.**
- **doc → Day of last change.** Das Datum, an dem das Passwort das letzte Mal geändert wurde. Es ist als Anzahl der Tage seit dem 1. Januar 1970 ausgedrückt.
- **minimum → minimale Anzahl der Tage, die das Passwort gültig ist.** Der Wert 0 bedeutet, dass das Passwort jederzeit geändert werden kann.
- **maximum → maximale Anzahl der Tage, die das Passwort gültig ist.** Ein leeres Feld hier bedeutet, dass es kein maximales Passwortalter, keinen Passwortwarnzeitraum und keinen Passwortinaktivitätszeitraum gibt.
- **passwortvorwarndauer → die Anzahl der Tage, die der Benutzer vor dem Ablauf des Passworts gewarnt wird**
- **deaktiv → die Anzahl der Tage, die das Passwort nach Ablauf von "maximum" noch gültig ist.** Danach kann sich der Benutzer nicht mehr anmelden. Kein Eintrag in diesem Feld bedeutet, dass das Konto sofort gesperrt wird, nachdem das Passwort abläuft.
- **deaktiv seit → ab diesem Tag (gezählt ab dem 1.1.1970) ist dieser Account gesperrt.** Kein Eintrag in diesem Feld bedeutet, dass das Konto nicht deaktiviert ist.

Zeile der Datei zu einem bestimmten Nutzer ausgeben: `sudo cat /etc/shadow | grep labadmin`

- (`grep` filtert die Ausgabe nach Schlüsselwort, Regex oder Muster) oder `chage -l labadmin`

7. Rechtemanagement

7.1. Rechte

Rechte anzeigen: `ls -ld`

Folgende Buchstabe beschreiben die Rechte einer Datei:

	Bezeichnung	**Effekt auf Datei**	**Effekt auf Verzeichnis**
`r`	read	Kann zum lesen geöffnet werden aber nicht verändert	Inhalt eines Verzeichnis kann mit ls angezeigt werden
`w`	write	Inhalt kann verändert werden Dateiname kann nicht verändert werden Datei kann nicht gelöscht werden	Dateien & Verzeichnise können gelöscht, umbenannt und neu erstellt werden (rwx notwendig)
`x`	execute	Datei kann ausgeführt werden (Bei Shell Skripten und Binärdaten wichtig)	Benutzer kann mit dem cd Befehl ins Verzeichnis wechseln
`-`	kein Recht	kein Recht	kein Recht

Einträge löschen, umbenennen oder erstellen –> alle drei Rechte (r,w,e) müssen vergeben sein

7.2. Zuordnung der Rechte

Berechtigungszuordnungen:

1. Besitzer/Eigentümer einer Datei (user)
2. Besitzende Gruppe (group)
3. Rest der Welt (other)

- jeder dieser Zuordnungen kann die Rechte r, w, x zugeordnet werden
- ein Anwender kann aber immer nur die Rechte über **eine** der Zuordnungen erhalten
- mind. Besitzer eine Datei –> user
- Mitglieder der besitzenden Gruppe –> group
- weder noch -> other
- **keine Vererbung von Rechten**

7.3. Beispiele

```BASH
-rw-rw-r--. 1 labadmin root 0 Mar 25 08:06 admin-file
```

Teil	Bedeutung
`-rw-rw-r--.`	Dateityp und Berechtigungen
`1`	Anzahl der Hardlinks
`labadmin`	Besitzer (User)
`root`	Gruppe (Group)
`0`	Dateigrösse (in Bytes)
`Mar 25 08:06`	Änderungsdatum
`admin-file`	Dateiname

Bereich	Bedeutung
`-`	Typ: normale Datei
`rw-`	Besitzer (labadmin) darf lesen und schreiben
`rw-`	Gruppe (root) darf lesen und schreiben
`r--`	Andere dürfen nur lesen

Die Datei gehört dem Benutzer **labadmin** und der Gruppe **root**. Alle dürfen lesen, aber nur der Besitzer und Gruppenmitglieder dürfen schreiben. Niemand darf sie ausführen.

```BASH
drwxr-xr-x. 4 labstudent labstudent 103 Mar 25 08:08 .
```

Teil	Bedeutung
`drwxr-xr-x.`	Dateityp und Berechtigungen
`4`	Anzahl der Links (Unterverzeichnisse + Elternverzeichnis + . + ..)
`labstudent`	Besitzer (User)
`labstudent`	Gruppe (Group)
`103`	Dateigrösse (in Bytes)
`Mar 25 08:08`	Änderungsdatum
`.`	Verzeichnis selbst (aktuelles Verzeichnis)

Bereich	Bedeutung
`d`	Typ: Verzeichnis
`rwx`	Besitzer darf lesen, schreiben, hineinwechseln
`r-x`	Gruppe darf lesen und hineinwechseln, aber nicht schreiben
`r-x`	Andere dürfen lesen und hineinwechseln, aber nicht schreiben

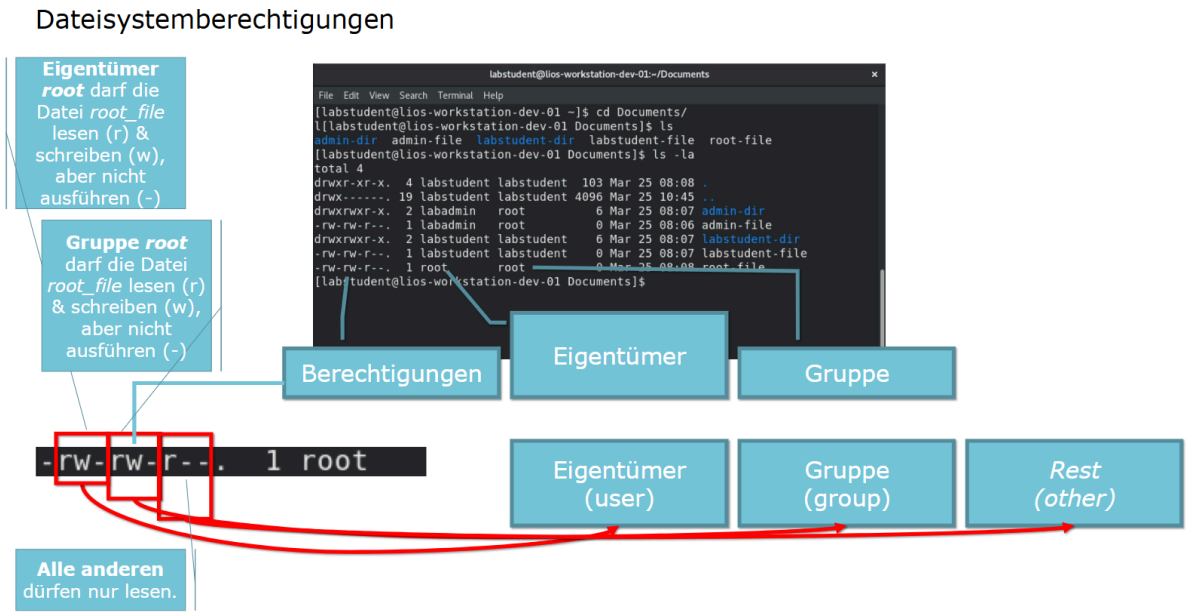
Das Verzeichnis gehört **labstudent**. Andere Benutzer können hineinschauen (`ls`) und wechseln (`cd`), aber keine Dateien darin ändern oder löschen.

7.4. Gängige Einstellungen

- Lese und Ausführungsberechtigungen für Verzeichnisse
  - Dateinamen und Dateiinhalte können angezeigt werden
- Nur Lesezugriff auf Verzeichnis

- Nur Dateinamen können angezeigt werden
- Nur Ausführzugriff auf eine Verzeichnis
  - Dateinamen können nicht aufgelistet werden, jedoch kann der Inhalt einer bekannten Datei angezeigt werden

7.5. Dateisystemberechtigungen



Recht	Binärwert	Oktalwert
read	2^2	4
write	2^1	2
execute	2^0	1

Zeichen	Bedeutung
-	normale Datei
d	Verzeichnis (directory)
l	symbolischer Link
b,s,p	b=blockorientierte Gerätedatei, s=Unix-Domainsocket, p=named pipe-Datei, c=zeichenorientierte Gerätedatei

Darstellung	Oktalschreibweise	Bedeutung
`[r--------]`	400	read (Leserecht für Eigentümer)
`[-w-------]`	200	write (Schreibrecht für Eigentümer)
`[--x------]`	100	execute (Ausführrecht für Eigentümer)
`[rwx------]`	700	read, write, execute (Lese-, Schreib- und Ausführungsrecht für Eigentümer)
`[---r-----]`	040	read (Leserecht für Gruppe)
`[----w----]`	020	write (Schreibrecht für Gruppe)
`[-----x---]`	010	execute (Ausführrecht für Gruppe)
`[---rwx---]`	070	read, write, execute für Gruppe
`[------r--]`	004	read (Leserecht für alle anderen Benutzer)
`[-------w-]`	002	write (Schreibrecht für alle anderen Benutzer)
`[--------x]`	001	execute (Ausführrecht für alle anderen Benutzer)
`[------rwx]`	007	read, write, execute für alle anderen Benutzer

7.6. Rechte ändern

Rechte können vom **root Benutzer und dem Besitzer einer Datei geändert** werden

Befehl: `chmod`

Chmod hat zwei Modi:

- **Relative Vergabe**: derzeitige Berechtigung wird geändert
- **Absolute Vergabe**: kompletter Berechtigungssatz wird für alle Zuordnungen neu erstellt

chmod [Wer][Wie][Was] file **Relativ**

- [Wer] **u, g, o, a**: user, group, other, all
- [Wie] **+, -, =**: hinzufügen, weg, setzen
- [Was] **r, w, x**: lesen, schreiben, ausführen

chmod ### file **Absolut**

- Eigentümer → Gruppe → Rest
- 4 (read), 2 (write), 1 (execute)
- Werte werden addiert (z.B. 4+2=6)

Beispiel relativ:

```sh
chmod u+x datei1     # Änderung: Eigentümer erhält Ausführrechte
chmod g-r,o-r datei1 # Änderung: Gruppe und Rest bekommen leserechte entzogen
chmod a+r datei1     # Änderung: Alle erhalten Leserechte
chmod a+rx datei1    # Änderung: Alle erhalten Lese und Ausführrechte
```

Beispiel absolut:

```sh
chmod 600 datei1 # Festelegung: Nur der Eigentümer erhält Lese und Schreibrechte
```

7.7. Ändern des Eigentümers

Der User der eine Datei anlegt ist ihr Eigentümer -> Das kann durch root geändert werden

Befehl: `chown Eigentümer:Gruppeneigentümerschaft file/directory`

Beispiel

```sh
chown :wheel labstuden-file # Gruppeneigentümer ist nun die wheel Gruppe
chown labstudent: labstuden-file # Eigentümer ist nun labstudent
chown root:root labstuden-file # Eigentümer ist die root Gruppe & der root User
```

7.7.1. chown bei symbolische Links

Änderungen des Eigentums von symbolischen Links -> Eigentum der Datei auf die der Link verweist wird geändert

Verhalten kann mit `-h` überschrieben werden

7.7.2. Spezielle Berechtigungen - Special Bits

Es gibt einen vierte Berechtigungstyp

- In Oktalschreibweise eine vierte Ziffer, z.B: 2700 (2 Wäre Special Bit)

Recht	Binärwert	Oktalwert	Beschreibung	Bemerkung
SUID	2^2	4	Temporäre Eigentümerschaft für den Aufrufer (Als würde vom root ausgeführt)	nur sinnvoll auf ausführbaren Dateien, z.B Passwort ändern (ändern der globalen /etc/shadow Datei) oder shutdown
SGID / GUID	2^1	2	Vererbung der Gruppeneigentümerschaft des Verzeichnisses auf neue Dateien und Verzeichnisse	sinnvoll bei geshartend Folders -> Gruppe des gesharten Folders wird automatisch übernommen
Sticky Bit	2^0	1	Nur noch Besitzer einer Datei kann diese löschen	Nur sinnvoll bei Verzeichnissen

Wichtig für die Anzeige (`ls -l`):

- Kleines `s` / `t` : Bit gesetzt und Ausführrecht (`x`) vorhanden.
- Grosses `S` / `T` : Bit gesetzt, aber Ausführrecht (`x`) fehlt (oft ein Konfigurationsfehler).

SUID (Set User ID) - Bitwert 4

- Wenn eine Datei mit gesetztem **SUID-Bit** ausgeführt wird, läuft sie **mit den Rechten des Dateieigentümers**, **nicht** mit den Rechten des Benutzers, der sie startet.
- Bsp. `-rwsr-xr-x 1 root root /usr/bin/passwd`
  - Jeder Benutzer darf `/usr/bin/passwd` ausführen,
  - aber der Befehl läuft mit den **Rechten des Besitzers (root)**,
  - so kann der Nutzer sein Passwort in `/etc/shadow` ändern – obwohl er dort **keine Schreibrechte** hat.
- Nur sinnvoll für **ausführbare Programme (Binärdateien)**.
- Bei Skripten oder normalen Dateien wäre das **unsicher oder sinnlos**.

```sh
# relativ
chmod u+s datei
# absolut
chmod 4755 datei
```

SGID / GUID (set Group ID) - Bitwert 2

- Vererbt **Gruppenzugehörigkeit** auf neue Dateien und Unterverzeichnisse.
- Bsp. `drwxrwsr-x  labstudent  projektgruppe  /srv/projekt`
  - Das **„s“** im Gruppenblock zeigt das **SGID-Bit**.
  - Alle neuen Dateien oder Unterverzeichnisse in `/srv/projekt` werden **automatisch der Gruppe „projektgruppe“** zugeordnet – **egal**, welcher Benutzer sie erstellt.
- praktisch für z.B. Teamverzeichnisse

```sh
# relativ
chmod g+s verzeichnis
# absolut
chmod 2770 verzeichnis
```

Sticky Bit - Bitwert 1

- Nur der **Dateibesitzer** (oder root) darf eine Datei löschen, auch wenn andere im Verzeichnis Schreibrechte haben.
- Bsp. `drwxrwxrwt  10 root  tmp  /tmp` (t am Ende zeigt das Sticky Bit)
  - Jeder darf in `/tmp` Dateien erstellen (da `rwx` für alle gilt),
  - aber nur der **Besitzer seiner eigenen Dateien** darf sie wieder löschen.
- Das schützt vor dem versehentlichen oder absichtlichen Löschen fremder Dateien in gemeinsam genutzten Verzeichnissen.

```sh
# relativ
chmod o+t verzeichnis
# absolut
chmod 1777 verzeichnis
```

7.8. Default Dateiberechtigungen

Initiale Berechtigung: Default Systemeinstellungen + umask

Default-Systemeinstellungen:

- Datei: `0666` `-rw-rw-rw-`
- Verzeichnisse: `0777` `drwxrwxrwx`

umask:

- Schränkt Default Berechtigungen ein, bzw. löscht die entsprechenden Bits der Default-Einstellungen
- umask ist oktale Bitmaske, d.h. Default-Systemberechtigungen werden von umask weiter eingeschränkt
- Ein Bit in der umask, löscht das entsprechende Bit der Default-Systemeinstellungen

Beispiel:

```
umask 0002 -> neue Datei 0666-0002 = 0664 -rw-rw-r--
umask 0077 -> neues Verzeichnis 0777-0077 = 0700 drwx------
```

umask ist **terminal-spezifisch** (Session-spezifisch), daher meist in einer User-spezifischen Konfigurationsdatei gesetzt, z.B:

- `~/.bashrc`
- `~/.profile`
- `~/.zshrc`

7.9. Erstellen eines privaten Ordners

1. Ordner in einem Verzeichnis auf das man Zugriff hat erstellen
   `sudo mkdir private_daten`
2. Eigentümerschaft und Gruppeneigentümerschaft des neuen Verzeichnisses auf sich und seine primäre Gruppe ändern
   `sudo chown labstudent:labstudent private_daten`
3. Berechtigungen so setzten, dass man nur selber Lese-, Schrieb- und Execute-Rechte auf dises Verzeichnis hat
   `sudo chmod 700 private_daten/`

Wichtig: Ein Verzeichnis und dessen Inhalte sind in Linux nur dann lesbar, falls das Verzeichnis und dessen Eltern (höhere Verzeichnisse) das execute-Recht für den jeweiligen Nutzer haben.

7.10. Erstellen eines gemeinsam genutzten Ordners

1. Ordner in einem Verzeichnis auf das alle Teilnehmer mind. execute-Zugriffe haben (d.h. sie können mit `cd` wechseln)
   `sudo mkdir sharing`
2. Eigentümerschaft auf sich, Gruppeneigentümerschaft auf eine Gruppe zu der die Teilnehmer gehören ändern.
   `sudo chown labstudent:wheel sharing`
3. Berechtingen genau so setzen, dass nur der Eigentümer und die Eigentümergruppe Lese-, Schreib- und Execute-Rechte auf das Verzeichnis haben.
   `sudo chmod 770 sharing`
4. GUID-Bit auf das Verzeichnis so setzen, dass künftige neue Dateien die Gruppe des Verzeichnisses erben
   `sudo chmod g+s ../sharing`

8. Logdateien

8.1. Systemprotokollierung

- für Fehlersuche und System-Auditing -> Ereignisse von Prozessen und Betriebssystem-Kernel werden in Protokolle aufgezeichnet
- viele Systeme zeichnen Ereignisprotokolle in Textdateien auf
  - im Verzeichnis `/var/log` gespeichert
- können in einem Textdienstprogramm wie `less` und `tail` untersucht werden

Generell umfasst Protokollierungssystem die Dienste:

`systemd-journald`

- zentrales Element der Ereignisprotokollierungsarchitektur
- Ereignismeldungen von vielen Quellen, z.B.
  - Kernel
  - Ausgaben aus den frühen Stadien des Systembootvorgangs
  - Standardausgaben und Standardfehler von Diensten bei ihrem Start und ihrer Ausführung
  - Syslog-Ereignisse
- Speicherung in Dateisystem (bleibt nach Bootvorgängen nicht bestehen): `/run/log`

`rsyslog`

- liest Syslog-Meldungen, die von `systemd-journald` empfangen werden,
- sortiert die Syslog-Meldungen,
- schreibt Syslog-Meldungen in Protokolldateien
- bleiben nach Bootvorgängen in `/var/log` erhalten

8.2. Ausgewählte Systemprotokolldateien

Protokolldatei	Beschreibung und Inhalt
`/var/log/messages`	• allg. Meldungen und Informationen zum System • im Grunde: Datenprotokoll aller Aktivitäten im gesamten System • ohne Meldungen zur Authentifizierung, E-Mail Verarbeitung, Ausführung terminierte Jobs Debian/Ubuntu: `/var/log/syslog`
`/var/log/secure`	• Authentifizierungsprotokolle (erfolgreiche und fehlgeschlagene) Debian/Ubuntu in `/var/log/auth.log`. Redhat/CentOS in `/var/log/secure`
`var/log/boot.log`	• Startmeldungen und Startinformationen • Konsolenmeldungen im Zusammenhang mit dem Systemstart, nicht von Syslog.
`/var/log/cron`	Speichert Nachrichten zu wiederkehrende Aufgaben und automatisierten Jobs: Start, Ende und Meldungen.
`/var/log/maillog`	Für Mailserver-Protokolle, praktisch für Postfix-, SMTPD- oder E-Mail- bezogene Dienstinformationen, die auf dem Server ausgeführt werden.
`/var/log/mysql` oder `/var/log/httpd`	Einige Prozesse und Dienste legen eigene Unterverzeichnisse an und speichern ihre Logs in Unterverzeichnissen.

8.3. Aufbau einer Syslog-Meldung

Eine Syslog-Meldung besteht aus drei Komponenten:

1. Selektor
2. Log-Header
3. eigentlicher Inhalt

8.3.1. Selektor

Selektor: Ganzzahlt, die aus zwei Teilen besteht

- Severity-Feld (Schwere, Heftigkeit)
- Facility-Feld

Severity (Schwere	Beschreibung
0 Emergency	Systemkritisch. Beeinflusst das System, sodass das System nicht mehr verwendet werden kann.
1 Alert	Fehler, die behoben werden müssen.
2 Critical	Schwerer Fehler.
3 Error	Es lag ein Fehler vor.
4 Warning	Warnung bei der Nutzung von einem Dienst oder systemweit.
5 Notice	Auftreten eines Ereignisses.
6 Informational	Information für die Nutzung.
7 Debug	Debugging-Meldung über den Ablauf eines Programms oder Dienst.

Facility-Feld

- kern, 0
- user, 1
- mail, 2
- daemon, 3
- auth, 4
- syslog, 5
- lpr, 6
- news, 7
- uucp, 8
- cron, 9
- authpriv, 10
- ftp, 11

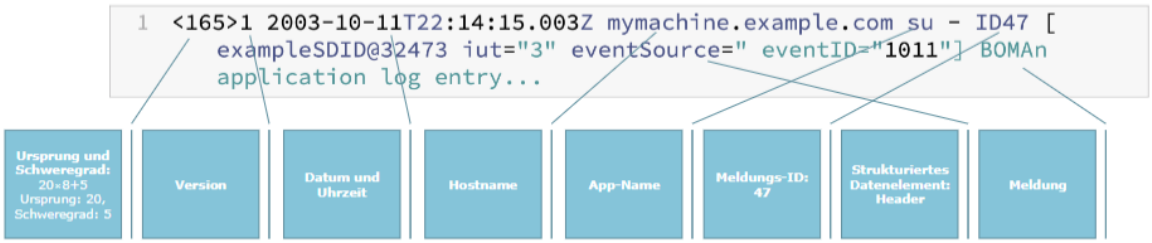
Berechnung **Selektor**: `Facility-Feld * 8 + Severity`

1. Facility-Feld mit 8 multiplizieren
2. numerische Wert der Schwere (Severity-Feld) addieren

8.3.2. Log-Header

Log-Header

- Zeitstempel und Name / Hostname des Absenders



8.4. Konfiguration von rsyslog

- `rsyslog` gruppiert Meldungen nach dem Selektor mit Facility-Feld (Ursprung) und Severity-Feld
- Speicherung von Meldungen in mehreren Protokolldateien -> wenn Meldung mit mehr als einer Regel in `rsyslog.conf` übereinstimmt
- `logrotate` rotiert die Protokolldateien so, dass sie nicht zu viel Speicherplatz in dem Dateisystem beanspruchen
  - Ab einer Grösse oder Alter werden Log- Dateien umbenannt, mit Zeitstempel versehen – und später gelöscht.

```sh
# Eigene Konfiguration
/etc/rsyslog.d/your-custom-config.conf

# Hauptkonfiguration
/etc/rsyslog.conf
```

8.5. Nächste Generation von Logging mit `Systemd`

`Systemd-journald`

- speichert Protokolldaten in einer strukturierten, indizierten Binärdatei (Bezeichnung: Journal)
- Daten enthalten zusätzliche Informationen zum Protokollereignis
- bei Syslog-Ereignissen enthalten die Daten z.B. die Facility und die Priorität
- Ort: `/run/log/journal`
- bei Bedarf mit `rsyslog` wegspeichern, wird nur bis zum nächsten Server-Bootvorgang gespeichert

8.5.1. Beispiele

Befehl	Beschreibung
`journalctl`	Zeigt alle Logeinträge des systemd-Journal-Dienstes an, der System- und Anwendungslogs enthält.
`journalctl -g "Ingwer-Shot"`	Durchsucht die System-Logs nach einem bestimmten Begriff oder einem regulären Ausdruck (Regex). (g = grep)
`journalctl -n 20`	Zeigt die letzten 20 Logeinträge an.
`journalctl -f`	Zeigt die neuesten Logeinträge in Echtzeit (ähnlich `tail -f`) an.
`journalctl --reverse`	Sortiert die Ausgabe umgekehrt, sodass die neuesten Nachrichten zuoberst stehen.
`journalctl -p err`	Zeigt nur Logeinträge mit der Priorität `err` (Error) und höher an.
`journalctl -p warning`	Filtert nach Warnungen und kritischeren Meldungen.
`journalctl -u cups*`	Filtert nach einer Systemd-Unit (Kurzform für `--unit`). Unterstützt Wildcards.
`journalctl --since today`	Zeigt alle Logeinträge ab Mitternacht des aktuellen Tages an.
`journalctl --since "-20min"`	Zeigt alle Logeinträge der letzten 20 Minuten an.
`journalctl --since=yesterday --until=now`	Zeigt alle Meldungen von gestern bis zum aktuellen Zeitpunkt an.
`journalctl -o verbose`	Zeigt Logeinträge im ausführlichen Format mit detaillierten Metadaten an.
`journalctl _SYSTEMD_UNIT=sshd.service _PID=1182`	Filtert explizit nach Systemd-Einheit und Prozess-ID (Lange Schreibweise).

8.5.2. `Systemd-Logs` per `rsyslog` wegschreiben

Parameter `Storage`

- in Datei `/etc/systemd/journald.conf`
- definiert, ob Systemjournale persistent oder volatil beim Booten gespeichert werden sollen
- `persisten, volatile` auf `auto` oder `none` festlegen
  - danach deamon neustartern `systemctl restart systemd-journald`
- Vorteil: historische Daten sind beim Booten sofort verfügbar
- Standard: Journal darf nicht grösser sein als 10% des Dateisystems, in dem es sich befindet oder es darf nur so viel Speicher brauchen, dass mehr als 15% des Dateisystems frei bleiben
  - ist ein integrierter Protokoll-Rotationsmechanismus der monatlich ausgelöst wird

8.6. Prozesse

8.6.1. Definition eines Prozesses

DEFINITION: Computerprogramm, welches gerade ausgeführt wird

Prozess = Computerprogramm, welches gerade ausgeführt wird
- Ein Computerprogramm bewirkt nichts, wenn es nicht gestartet wurde.
- Start eines Programms -> Kopie des Programms wird in den Arbeitsspeicher geladen
- Diese Instanz wird so zu einem (Betriebssystem-)Prozess, der zum Ablauf einem Prozessorkern zugeordnet werden muss.

Prozess besteht zusätzlich aus:
- Ressourcenzuteilung: Adressraum von zugewiesenem Arbeitsspeicher
- Sicherheitseigenschaften: wie Eigentümerinformationen und -berechtigungen
- Verwaltungsinformationen: Prozessstatus

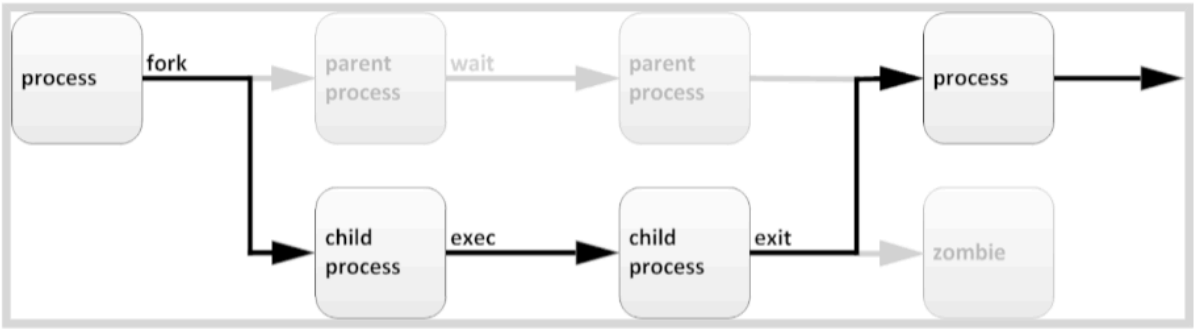
Umgebung eines Prozesses beinhaltet folgendes:
- lokale und globale Variablen
- aktuellen Umgebungskontext, z.B. Interpreter, Eltern-Prozess
- zugewiesene Systemressourcen wie Dateideskriptoren und Netzwerkports

8.6.2. Ablauf: Starten eines neuen Prozess
- Starten eines Prozesses wird immer durch einen anderen Prozess (Elternprozess) ausgelöst
  - neuer Prozess (Kindprozess) erbt Rechte und Umgebung
- `systemd` : System- und Dienstmanager in Rocky Linux und RedHat Enterprise Linux
  - ist der erste Prozess der beim Booten als `PID 1` ausgeführt wird
  - alle weiteren Prozesse erben von ihm die Umgebung und Sicherheitseinstellungen

Starten eines neuen Prozess:
1. (Übergeordneter) Prozess erstellt eine neue (untergeordnete) Prozessstruktur von sich selbst **`fork()`** : erstellt Duplikat es aktuellen Prozesses, der fast identisch ist Idee: Sicherheitseigenschaften und Umgebung vererben
2. jeder Prozess erhält eine eindeutige Prozess-ID (PID) übergeordnete Prozess ist via PPID abfragbar
3. neue Prozess ersetzt seinen Programmcode mit **`exec()`** ersetzt den aktuellen Prozess durch ein neues Programm, lädt das Programm in den aktuellen Prozessraum und führt es vom Einstiegspunkt aus

Daher werden `fork()` und `exec()` oft nacheinander verwendet, um ein neues Programm als Kind eines aktuellen Prozesses zu starten. Shells tun dies normalerweise, wenn sie versuchen, ein Programm auszuführen - die Shell verzweigt sich (fork), dann lädt das Kind das neue Programm in den Speicher und führt es aus (exec).

8.6.3. Prozess-Lebenszyklus

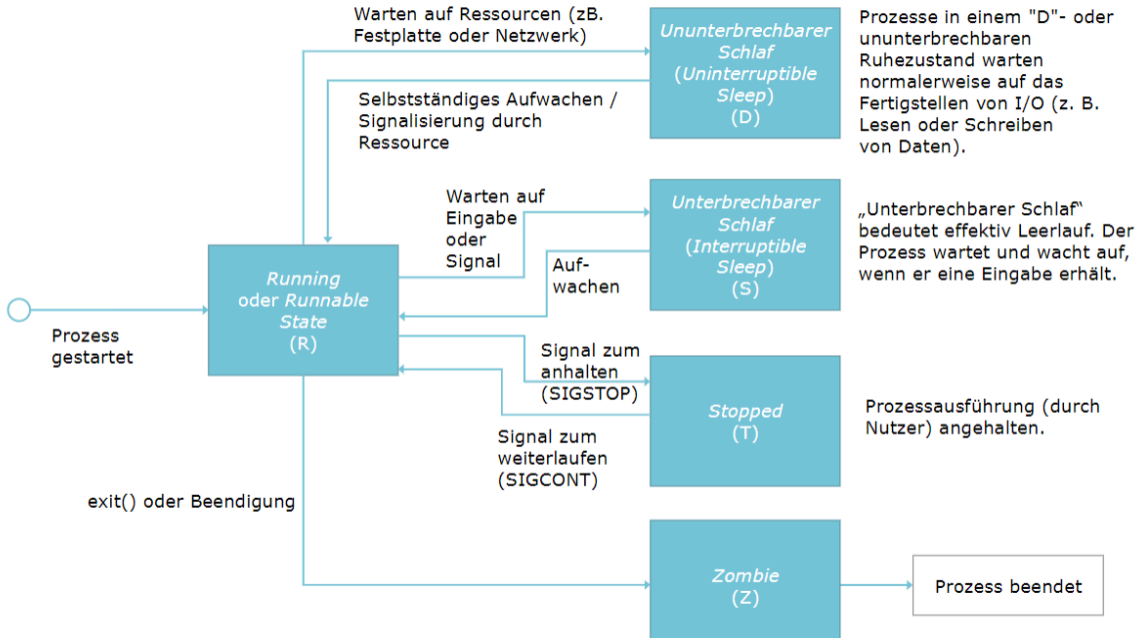


- `fork` : ein untergeordneter Prozess erbt
  - Sicherheitsmerkmale
  - alte und aktuelle Dateideskriptoren

- Port- und Ressourcenberechtigungen
- Umgebungsvariablen
- Programmcode

- `exec` : untergeordneter Prozess kann dann eigenen Programmcode ausführen
- Normalfall: übergeordneter Prozess im Ruhezustand, untergeordnete Prozess wird ausgeführt
  - `wait` : Anforderung, dass er beim Abschluss des untergeordneten Prozesses benachrichtigt wird

8.6.4. Prozess-Status



8.6.4.1. Zombie

Prozess wird beendet:

- Elternprozess kann vom OS erfragen, auf welche Art der Kindprozess beendet wurde
  - erfolgreich, mit Fehler, abgestürzt, abgebrochen etc.
- Eintrag bleibt in der Prozesstabelle bestehen, bis der Elternprozess diese Abfrage durchgeführt hat
  - egal ob diese Information benötigt wird oder nicht
- Bis zu diese Abfrage hat der Kindprozess den Zustand `Zombie`
  - Prozess belegt keinen Arbeitsspeicher mehr, bis auf den Eintrag in der Prozesstabelle des Kernels
  - verbraucht keine Rechenzeit
  - aber behält seine PID -> darf noch nicht für andere Prozesse wiederverwendet werden

Verwaiste Prozesse:

- Elternprozess wird beendet -> alle Kindsprozesse sind verwaist
- Kindsprozesse werden per Definition vom Prozess mit PID 1 adoptiert (erhalten diesen als Elternprozess)
- wenn keine Statusabfrage durchgeführt wird, bleibt der Kindprozess im Zombie-Zustand in der Prozesstabelle

Name	Flag	Statusname und Beschreibung definiert durch Kernel
Running	R	Der Prozess verarbeitet im Moment gerade Daten.
Sleeping	S	Der Prozess wartet auf eine Bedingung wie eine Hardware-Anforderung, den Zugang zu Systemressourcen oder ein Signal.
Deep Sleeping	D	Der Prozess befindet sich in einem **ununterbrechbaren Schlaf**.
	K	Dieser Status entspricht dem nicht unterbrechbaren Status **D**. Eine wartende Aufgabe kann hier jedoch auf das Signal zur vollständigen Beendigung reagieren.
	I	Eine Untermenge des Status **D**. Wird aber für Kernel-Threads verwendet.
Stopped	T	Der Prozess wurde **angehalten** (ausgesetzt) oder wird gedebugged.
Zombie	Z	Ein „toter“ Prozess. Er hat seine Arbeit beendet, belegt aber noch einen Platz in der Prozesstabelle, weil sein „Eltern-Prozess“ noch nicht die Bestätigung abgeholt hat, dass er fertig ist.
Dead / Exit	X	Der finale Zustand. Sobald der übergeordnete Prozess alles aufgeräumt hat, wird die Struktur gelöscht und der Prozess ist endgültig aus dem System verschwunden.

8.6.5. Linux-Prozessablauf

- Ein Prozess kann nicht unbegrenzt ausgeführt werden (geht zu lasten anderer laufenden Prozesse und verhindert Multitasking)
- Ende der Prozesssequenzierung ist wie folgt:
  - schliessen der geöffneten Dateien
  - Freigabe des belegten Speichers
  - Senden eines Signals an die übergeordneten und untergeordneten Prozesse
- Elternteil stirbt -> Kinder sind Orphans (Weisen)
  - werden vom Init-Prozess übernommen

Gerade bei der Fehlersuche im System ist es wichtig, die Prozesse und deren Status zu kennen. Insbesondere ist es wichtig zu wissen, wann ein Prozess auf externe Ressourcen wie Platten oder Netzwerk wartet (Stauts: D) oder wann ein Prozess im Leerlauf ist.

8.6.6. ps aux & top

Befehl	Beschreibung
`ps`	Zeigt eine Liste der aktuell laufenden Prozesse im aktuellen Terminal an.
`ps f`	Zeigt eine baumartige Darstellung der aktuell laufenden Prozesse im aktuellen Terminal an, wodurch die Eltern-Kind-Beziehungen sichtbar werden. BSD-Syntax. `f` : forrest, baumartige Darstellung
`ps aux`	Zeigt alle laufenden Prozesse an, inklusive Benutzerinformationen, CPU- und Speicherverbrauch, veralteter BSD-Syntax aber sehr gebräuchlich. `a` : Zeigt Prozesse von allen Nutzern `u` : Zeigt den Benutzer/Eigentümer des jeweiligen Prozesses `x` : Zeigt auch Prozesse, die nicht vom aktuellen Terminal abhängen, an.
`ps lax`	Zeigt alle Prozesse mit zusätzlichen Details und erweitertem Informationen, darunter Priorität und Status, veralteter BSD-Syntax. `l` : long -> +PID (Prozess-ID), +PPID (Elternprozess-ID), Scheduling-Priorität, etc. `a` : Zeigt Prozesse von allen Nutzern `x` : Zeigt auch Prozesse, die nicht vom aktuellen Terminal abhängen, an.
`ps -ef`	**Linux-Syntax**: Zeigt alle Prozesse in einem vollständigen Format an, inklusive vollständiger Kommandozeileninformationen. - `e` : Zeigt alle Prozesse an, unabhängig von ihrem Terminal oder Benutzer. Dies schliesst sowohl Systemprozesse als auch Benutzerprozesse ein. - `f` : Zeigt die Prozesse im „vollständigen Format“ an, das zusätzliche Informationen wie Benutzername, PID (Prozess-ID), PPID (Elternprozess-ID), Startzeit und den vollständigen Befehl (Kommandozeile) enthält.
`top`	Zeigt die laufenden Prozesse und Systemressourcen in Echtzeit an, ähnlich wie `ps aux`, jedoch live aktualisiert.

8.6.7. Jobs im Vorder- und Hintergrund

Prozesse können:

- im Vordergrund laufen -> Status R (running)
- im Hintergrund laufen -> Status R (running)
- gestoppt werden -> Status T (stopped) „freeze“

Befehl	Beschreibung
`<command>`	Startet den Befehl im **Vordergrund**. -> „Normalfall“
`<command> &`	Startet den Befehl im **Hintergrund** statt im Vordergrund. Der Prozess läuft im Hintergrund weiter, während der Nutzer weiterarbeiten kann.
STRG-Z (Tastatur-Eingabe während eines laufenden Prozesses im Vordergrund)	**Stoppt** einen aktuell laufenden Vordergrundprozess und setzt ihn in den **Hintergrund**, wo er pausiert bleibt. „freeze“. (Tastaturkombination für `kill -TSTP <PID>` siehe Abschnitt zu Signalen) Wird oft genutzt, wenn ein Prozess im Vordergrund gerade läuft, aber in den Hintergrund soll. „z. B. &-Zeichen beim Command vergessen…“ (Alternative zu: neues Terminal öffnen und `kill -TSTP <PID>` eingeben)
`jobs`	Zeigt eine Liste der aktuell laufenden Hintergrundprozesse im Terminal an.
`fg %1`	Bringt den angegebenen Hintergrundprozess aus `jobs` (z. B. `%1`) wieder in den Vordergrund und führt ihn weiter aus.
`fg`	Bringt den zuletzt in den Hintergrund verschobenen Prozess zurück in den Vordergrund und führt ihn weiter aus.
`bg %1`	Setzt den angegebenen gestoppten Prozess (z. B. `%1`) in den Hintergrund und führt ihn weiter aus.
`bg`	Setzt den zuletzt gestoppten Prozess (mit STRG-Z) in den Hintergrund und führt ihn dort weiter aus.

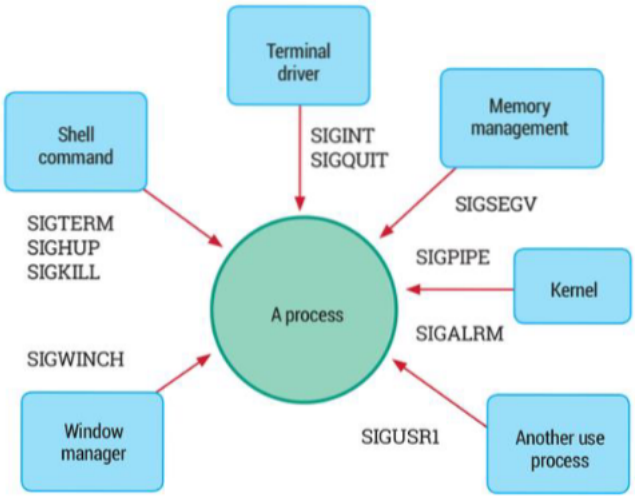
8.6.7.1. STRG-Z in der Praxis

Um einen laufenden Vordergrund-Prozess in den Hintergrund zu setzen, gibt es eine Kombination von Tastenkombinationen und dem Befehl `bg`.

1. **mit STRG-Z anhalten**: Zunächst muss der laufende Prozess mit der Tastenkombination STRG-Z angehalten werden. Das setzt den Prozess in den „stopped“ Zustand (T für „stopped“) und pausiert ihn freeze.
2. **mit `bg` in den Hintergrund setzen**: Anschliessend kann der pausierte Prozess mit dem Befehl bg im Hintergrund weiterlaufen. Dadurch ändert sich sein Status in „running“, und er läuft als Hintergrundprozess weiter. (`%1` -> falls es mehrere Prozesse gibt und man den spezifischen Prozess `%1` in den Hintergrund setzten möchte)

8.6.8. Steuern von Prozessen über Signale in Linux

- Signal = Software-Interrupt eines Prozesses
- Signale melden Ereignisse an ein ausgeführtes Programm
- Ereignisse, die ein Signal generieren, können durch
  - einen Fehler
  - ein externes Ereignis (eine I/O-Anforderung oder Zeitüberschreitung) oder
  - die explizite Verwendung eines Befehls, der ein Signal sendet, oder
  - durch eine Tastatureingabe ausgelöst werden.



Jedes Signal hat eine Standardaktion. Normalerweise ist das eine der folgenden:

- **Ende**: eendet ein Programm sofort (Englisch: Term)
- **Core**: Erstellt ein Speicherabbild (Speicherauszug), bevor das Programm beendet wird
- **Anhalten**: Unterbricht ein Programm (suspend) und wartet auf dessen Fortsetzung (resume) (Englisch: Stop)

Programme können durch Implementieren von Handler-Routinen, die die Standardaktion eines Signals ignorieren, ersetzen oder erweitern, vorbereitet werden, auf erwartete Ereignissignale zu reagieren.

8.6.8.1. Grundlegende Prozessverwaltungssignale

In der folgenden Tabelle sind die grundlegenden Signale aufgeführt, die Systemadministratoren für die Routine- Prozessverwaltung verwenden. Sie können auf Signale entweder mit ihrem Kurznamen (HUP) oder vollständigen Namen (SIGHUP) verweisen.

Name	Nr	Aktion	Standard	Bedeutung
SIGSTOP	19	Anhalten	POSIX	Der Prozess wurde angehalten (blockiert)
SIGTSTP	20	Anhalten	POSIX	Der Prozess wurde „von Hand“ durch den Benutzer angehalten (STRG-Z)
SIGCONT	18	Weiterlaufen	POSIX	Ein angehaltener Prozess soll weiterlaufen.
SIGINT	2	Ende	POSIX	Interrupt durch das Terminal oder den Benutzer (STRG-C)
SIGQUIT	3	Ende und Core	POSIX	Das Signal `quit` vom Terminal. Ähnlich zu SIGINT aber mit Speicherabbild (`STRG-\`)
SIGTERM	15	Ende	POSIX	Beendigung des Programms anfordern. Programme, die SIGTERM abfangen, bieten meistens einen »Soft Shutdown« an.
SIGKILL	9	Ende	POSIX	Sofortiges Beenden des Programmprozesses
SIGHUP	1	Ende	POSIX	Das (virtuelle) Terminal oder der Interpreter wurde geschlossen oder beendet

Signalnummern variieren auf den verschiedenen Linux-Plattformen. Die Signalnamen und deren Bedeutung sind aber standardisiert. -> Signalnamen merken, nicht Nummern

8.6.8.2. Befehle zum Senden von Signalen durch explizite Anforderungen

- Signal an den aktuellen Vordergrundprozess durch Tastatureingaben
  - Aussetzen/Anhalten: STRG+Z
  - Beenden: STRG+C oder `kill`
- `kill`
  - sendet ein Signal an einen Prozess über die PID
  - es können alle Signale gesendet werden, nicht nur die zum Beenden von Programmen
  - `kill -l` : Namen und Nummern aller verfügbaren Signale anzeigen

8.6.8.2.1. Befehl `kill`

```
[user@host ~]$ kill -l
1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL 5) SIGTRAP
10) SIGUSR1 15) SIGTERM 20) SIGTSTP 6) SIGABRT 11) SIGSEGV
16) SIGSTKFLT 17) SIGCHLD
      ...
```

```
[user@host ~]$ ps aux | grep job
5194 0.0 0.1 222448 2980 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/
   control job1
5199 0.0 0.1 222448 3132 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/
   control job2
5205 0.0 0.1 222448 3124 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/
   control job3
5430 0.0 0.0 221860 1096 pts/1 S+ 16:41 0:00 grep --color=auto job [
user@host ~]$ kill 5194
[user@host ~]$ ps aux | grep job
user 5199 0.0 0.1 222448 3132 pts/1 S 16:39 0:00 /bin/bash /home/ user/
   bin/control job2
user 5205 0.0 0.1 222448 3124 pts/1 S 16:39 0:00 /bin/bash /home/ user/
   bin/control job3
user 5783 0.0 0.0 221860 964 pts/1 S+ 16:43 0:00 grep --color=auto job
[1] Terminated control job1
```

```
[user@host ~]$ kill -9 5199
[user@host ~]$ ps aux | grep job
user 5205 0.0 0.1 222448 3124 pts/1 S 16:39 0:00 /bin/bash /home/ user/
   bin/control job3
user 5930 0.0 0.0 221860 1048 pts/1 S+ 16:44 0:00 grep --color=auto
job
[2]- Killed control job2
```

```
[user@host ~]$ kill -SIGTERM 5205
user 5986 0.0 0.0 221860 1048 pts/1 S+ 16:45 0:00 grep --color=auto
 job
[3]+  Terminated              control job3
```

8.6.8.2.2. Befehl `killall`

```
[user@host ~]$ ps aux | grep job
5194 0.0 0.1 222448 2980 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/
   control job1
5199 0.0 0.1 222448 3132 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/
   control job2
5205 0.0 0.1 222448 3124 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/
   control job3
5430 0.0 0.0 221860 1096 pts/1 S+ 16:41 0:00 grep --color=auto job

[user@host ~]$ killall control
[1] Terminated
[2]- Terminated
[3]+ Terminated

[user@host ~]$
```

8.6.8.2.3. Befehl `pkill`

- Signal an einen oder mehrere Prozesse zu senden, die den Auswahlkriterien entsprechen
- Auswahlkriterien können sein
  - Befehlsname
  - Prozess eines bestimmten Benutzers
  - alle systemweiten Prozesse

```
[user@host ~]$ ps aux | grep pkill
user 5992 0.0 0.1 222448 3040 pts/1 S 16:59 0:00 /bin/bash /home/ user/
    bin/control pkill1
user 5996 0.0 0.1 222448 3048 pts/1 S 16:59 0:00 /bin/bash /home/ user/
    bin/control pkill2
user 6004 0.0 0.1 222448 3048 pts/1 S 16:59 0:00 /bin/bash /home/ user/
    bin/control pkill3
[user@host ~]$ pkill control
[1] Terminated control pkill1
[2]- Terminated control pkill2
[user@host ~]$ ps aux | grep pkill
user 6219 0.0 0.0 221860 1052 pts/1 S+ 17:00 0:00 grep --color=auto
    pkill
[3]+ Terminated control pkill3
```

```
[user@host ~]$ ps aux | grep test
user 6281 0.0 0.1 222448 3012 pts/1 S 17:04 0:00 /bin/bash /home/ user/
    bin/control test1
user 6285 0.0 0.1 222448 3128 pts/1 S 17:04 0:00 /bin/bash /home/ user/
    bin/control test2
user 6292 0.0 0.1 222448 3064 pts/1 S 17:04 0:00 /bin/bash /home/ user/
    bin/control test3
user 6318 0.0 0.0 221860 1080 pts/1 S+ 17:04 0:00 grep --color=auto
    test
[user@host ~]$ pkill -U user
[user@host ~]$ ps aux | grep test
user 6870 0.0 0.0 221860 1048 pts/0 S+ 17:07 0:00 grep --color=auto
    test
[user@host ~]$
```

`SIGKILL` wird von Admins zu schnell verwendet –> führt immer zu Beendigung, Prozess hat nicht die Möglichkeit zur Selbstbereinigung. Immer zuerst `SIGTERM` zu senden und erst dann `SIGKILL`

8.7. Dienste

Wird als Daemon oder Services bezeichnet (Windows: Systemdienste)

Ein Dienst ist ein Prozess der im Hintergrund abläuft:
- Werden beim Systemstart automatisch ausgeführt
- Keine Nutzerinteraktion während des Betriebs
- Konfiguration über conf Dateine im `/etc`
- Namensschema: abschliessendes d (sshd, crond)

8.7.1. `Systemd`

Der systemd Daemon verwaltet den Startvorgang von Linux und startet die nötigen Dienste
- ist selbst ein Dienst und der erste Prozess der gestartet wird (PID 1)
- Standard-Initialisierungssystem seit Redhat/CentOS 7
- Systemressourcen, Server-Daemons und andere Prozesse können sowohl beim Bootvorgang als auch auf einem laufenden System aktiviert werden

Wichtige Funktionen:
- Parallerler Start von Systemdiensten beim Bootvorgang
- On-Demand Aktivierung von Daemons
- Verwaltung von Abhängigkeiten zwischen Diensten
- Mount Points können als systemd Ziele konfiguriert werden

-> Alle Operationen der Dienste haben ein Standard Timeout von 5 Minuten, damit sie nicht das Sytem einfrieren bei Fehlern

8.7.1.1. Service Units

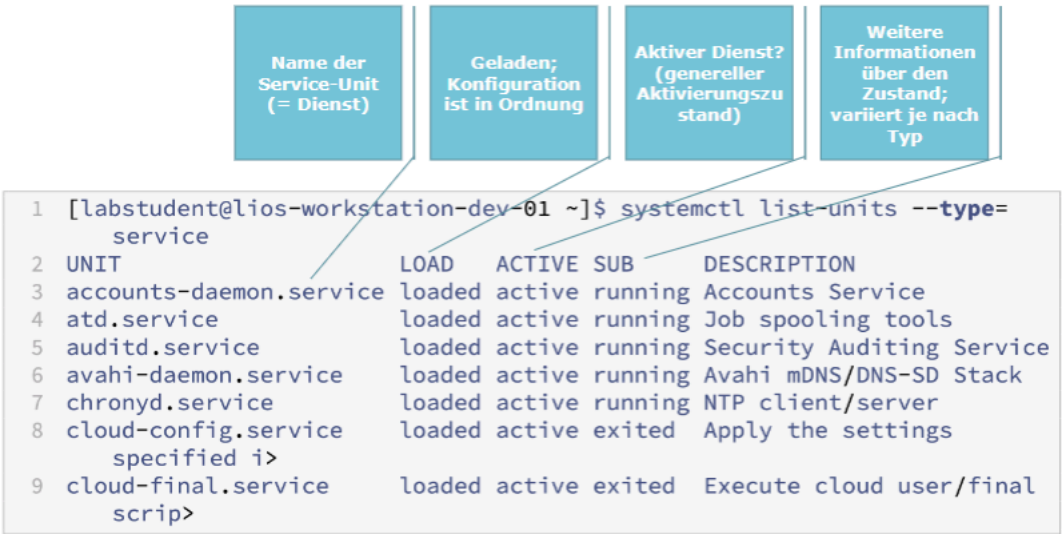
Verschiedene Arten von Units werden verwendet um unterschiedliche Teile des Systems zu verwalten

- `systemd` führt das Konzept der Systemd-Units ein
- Systemd verwendet verschiedene Arten von Units
  - um unterschiedliche Aspekte des Systems zu verwalten
  - jede Unit hat eine spezifische Rolle oder Funktion

Typ	Dateierweiterung	Beschreibungen
Service-Unit	.service	Systemdienst
Target-Unit	.target	Gruppiert andere Units und dienen als Synchronisationspunkte während des Bootvorgangs
Timer-Unit	.timer	Ähnliche Funktionen wie Cron Jobs, Services können zu einem bestimmten Zeitpunkt/Intervall ausgelöst werden
Socket Units	.socket	Stellt Sockets für Systemdienste zur Verfügung (z.B. Netzwerkverbindung)
Device Units	.device	Repräsentiert vom Kernel erkannte Geräte
Mount Units	.mount	Bindet Dateisysteme ein. Wird beim Systemstart benltigt um Dateisysteme einzubinden
Automount Units	.automount	Automount Punkte für die Mount Units. Diese werden automatisch eingebunden
Swap Units	.swap	Verwaltet Swap Räume
Path Units	.path	Überwachen eine Datei oder ein Verzeichnis auf Änderungen, sie können verwendet werden um eine Service-unit zu triggern wenn eine Datei geändert wurde
Slice Units	.slice	Definiert hierarchise Gruppen von Ressourcen, die von Control Groups verwaltet werden

Typ	Dateierweiterung	Beschreibungen
Scope Units	.scope	Werden für externe Prozesse verwenden die von einem anderen Prozess erstellt wurden und nicht durch systemd

Auflisten von aktiven Service Units: `systemctl list-units --type=service`



Auflisten von allen Units: `systemctl`
Auflisten aller Dienste die automatische gestartet werden beim Systemstart:
`systemctl list-unit-files --type=service`

8.7.1.2. Servicestatus

Anzeigen des Service Status: `systemctl status sshd.service`

Befehl	Beschreibung
`systemctl is-active sshd.service`	Dienst generell aktiv
`systemctl is-enabled sshd.service`	Dienst startet beim Boot automatisch
`systemctl is-enabled sshd.service`	Gab es Fehler bei der Ausführung
`systemctl --failed --type=service`	Alle Dienste mit Fehlern auflisten

Generelle Befehle

Befehl	Beschreibung
`systemctl status UNIT`	Detailiete Statusinformaitonen anzeigen
`systemctl stop UNIT`	Service stoppen
`systemctl start UNIT`	Service start
`systemctl restart UNIT`	Service neustarten
`systemctl reload UNIT`	Neuladen der Konfigurationsdatei von einem Service
`systemctl mask UNIT`	Ausführen von einem Service unterbinden
`systemctl unmask UNIT`	Einen maskierte Service verfügbar machen
`systemctl enable UNIT`	Dienst konfigurieren damit er beim Systemstart ausgeführt wird
`systemctl disable UNIT`	Dienst konfigurieren damit er beim Systemstart **nicht** ausgeführt wird
`systemctl list-dependencies UNIT`	Dependencies für den Start des Services auflisten

9. Archivierung

Daten komprimieren und archivieren ist nützlich bei der Erstellung von **Datensicherungen** und um **Daten in einem Netzwerk zu übertragen**.

9.1. tar

tar (Tape ArchiveR) ist einer der ältesten und meistverwendeten Befehle zu Erstellen und Arbeiten mit Sicherungsarchiven

- strukturierte Folge von Dateidaten und Metadaten zu jeder Datei
  - enthält Index um Dateien einzeln zu extrahieren
- ermöglicht das Speichern auf mehreren aufeinanderfolgenden Medien
- Archiv kann mit Komprimierungsmethoden gzip, bzip2, oder xz komprimiert werden
- `tar`-Archiv = strukturierte Folge von Dateidaten mit Metadaten zu jeder Datei und einem Index, sodass einzelne Dateien extrahiert werden können
- `tar` ermöglicht das Speichern auf mehreren aufeinanderfolgenden Medien (Multi-Volume-Optionen)
- Archiv kann

`tar`: Packt Dateien zusammen und ruft einen Verpackungsalgorithmus wie z.B. `gzip` (Option `-z`) auf

9.1.1. Windows

- seit Windows 10 1803 wird `tar` mit installiert
- ältere Versionen können Dateien, die mit `tar` gepackt wurden nicht direkt entpacken bzw. öffnen
- zusätzliche Archivierungsprogramme (`7-Zip, TUGZip, IZArc`) nötig (gibt noch weitere)

9.1.2. Befehl

Operationen

Option	Beschreibung
`-c, --create`	Neues Archiv erstellen
`-x, --extract`	Aus vorhandenem Archiv extrahieren
`-t, --list`	Inhaltsverzeichnis eines Archivs auflisten

Optionen

Option	Beschreibung
`-v, --verbose`	Zeigt an welche Dateien archiviert oder extrahiert werden
`-f, --file=`	File-Name, auf diese Option muss der Dateiname des zu verwendenden Archiv verwendet werden
`-p, --preserve-permissions`	Berechtigungen für Dateien und Verzeichnisse beibehalten

Komprimierungsoptionen (in steigender Komprimierungsrate)

Option	Beschreibung
`-z, --gzip`	Komprimierung gzip verwenden (.tar.gz)
`-j, --bzip2`	Komprimierung bzip2 verwenden (.tar.bz2)
`-J, --xz`	Komprimierung xz verwenden (.tar.xz)

Beispiele Erstellen

```sh
tar -czf /root/etcbackup.tar.gz /etc
tar -cjf /root/logbackup.tar.bz2 /var/log
tar -cJf /root/sshconfig.tar.xz /etc/ssh
```

Beispiele Auslesen

```bash
tar -tzf /tmp/etc.tar.gz
```

Beispiel Extrahieren

```bash
tar -xfz /tmp/etc.tar.gz
```

9.2. Übertragung von Dateien

9.2.1. Secure Copy: scp

- scp ist Teil der OpenSSH Suite
- kopiert Dateien von einem Remote System auf das lokale System oder umgekehrt
- verwendent SSH-Server für die Authentifizierung
- verschlüsselt Daten bei der Übertragung

```sh
# Quelle lokal (2 Dateien: yum.conf & hpsts), Ziel remote
scp /etc/yum.conf /etc/hpsts remoteuser@remotehost:/home/remoteuser
# Quelle remote, Ziel lokal
scp remoteuser@remotehost:/etc/hostname /home/user
scp -r labstudent@workstation-010:~/Documents /workstation-backup #rekursiv Documents auf
labstudent nach /workstation-backup
```

9.2.2. FTP-Modus: sftp

- Dateien Interaktiv hoch- oder herunterladen
- Verwendet sicher Authentifizierung und verschlüsselt Datenübertragung

```sh
# Datei hochladen
sftp> mkdir hostbackup
sftp> cd hostbackup
sftp> put /etc/hosts
# Datei herunterladen
sftp> get /etc/yum.conf
sftp> exit
```

9.2.3. rsync

Minimiert kopierte Datenmenge in dem nur geänderte Teile der Dateien synchronisiert werden

-> sehr effiziente Kopie

Mit -a (Archivemodus) aktivierte Optionen

Option	Beschreibung
`-a`	Archivmodus, ist rekursiv (berücksichtigt auch Unterverzeichnisse und Dateien)
`-v`	verbose, zeigt welche Dateien kopiert wurden
`-r, --recursive`	Rekursives Synchronisieren
`-l, --links`	Synchronisiert symbolische Links
`-p, --perms`	Beibehalten von Berechtigungen
`-t, --times`	Beibehalten von Zeitstempeln
`-g, --group`	Beibehalten von Gruppeneigentümerschaften
`-o, --owner`	Beibehalten von Owner
`-D, --devices`	Synchronisieren von Gerätedateien

```sh
rsync -av /var/log labstundent@server-004:~/workstation-logs
```

10. Installation

10.1. Red Hat Subscription

- Registrierung des Systems im Red-Hat-Konto
- Zuweisung von Berechtigungen für Updates und Support
- Subscription Management:
  - Tools zum Verwalten von Produktsubskriptionen
  - Berechtigungen beziehen, Updates erhalten
  - Supportverträge und Subskriptionen nachverfolgen
- Kostenloser Developer-Zugang für Studierende
- PackageKit und yum:
  - Zugriff auf Softwarepakete und Updates
  - Nutzung des Red-Hat-Netzwerks zur Inhaltsverteilung

10.1.1. Ablauf

1. System registrieren und dem Red-Hat-Konto zuordnen
2. Subskription erstellen -> Updates und Supportstufen zuweisen
3. Repos aktivieren / deaktivieren -> Softwarepakete bereitstellen
4. Berechtigungen prüfen und nachverfolgen

10.1.2. Verwaltung über Kommandozeile

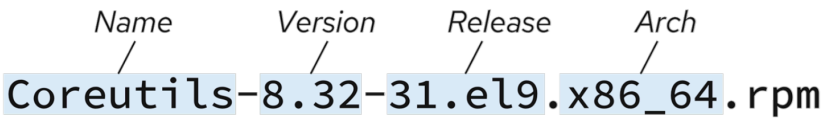
- `subscription-manager` dient zur Registrierung eines Systems ohne GUI
- Registrierung via: `subscription-manager register --username <name>`
- Automatische Zuordnung kompatibler Subscriptions
- Verfügbare Subskriptionen anzeigen mit: `subscription-manager list --available`
- Subscriptions automatisch zuordnen: `subscription-manager attach --auto`
- Bestimmte Subscription zuordnen (per Pool-ID): `subscription-manager attach --pool=<poolID>`
- Verbrauchte Subskriptionen anzeigen: `subscription-manager list --consumed`
- Registrierung eines Systems aufheben: `subscription-manager unregister`

10.2. RPM - Redhat Package Manager

- Früher Verteilung von Software über tgz-Archive
  - keine automatisierte Verwaltung
  - Abhängigketien wurden nicht aufgeloest
  - Warnungen/Hinweise in separaten Dateien
- 1995 für Red Hat Linux 2.0 entwickelt
- Anfänglich Perl-Skripte zur Paketverwaltung, heute in C implementiert
- Ziel von RPM:
  - Archivformat für Softwareinstallation und -aktualisierung
  - kryptographisch gesichert
  - enthält Skripte, Man-Pages und Binärdateien
- Eingesetzt u. a. in:
  - CentOS
  - openSUSE
  - Red Hat Enterprise Linux
  - AIX, Solaris (Unix-Systeme)

10.2.1. RPM-Paketdateien

Ein RPM-Paketname besteht aus: **Name-Version-Release.Architektur**



Name: Bezeichnung des Paketinhalts (z.B. coreutils)

Version: Versionsnummer der ursprünglichen Software (z.B. 8.32)

Release: Release-Nummer vom Paketbauer, Gibt Anzahl der Builds an, Enthält Distributionskennung (z. B. el9) -> 31.el9 = 31. Build für Enterprise Linux 9

Arch: Ziel-Prozessor-Architektur, für die das Paket kompiliert wurde (z.B. `x86_64` für 64-Bit-x86, `aarch64`)

10.2.2. Installation

- RPM-Pakete werden meist aus Repositories installiert (zentraler Speicher- und Verwaltungsort für Pakete).
- Für die Installation reicht der Paketname.
- Bei mehreren Versionen wählt RPM automatisch die höchste Versionsnummer.
- Bei mehreren Releases derselben Version installiert RPM das Release mit der höchsten Release-Nummer.

10.2.3. Befehle

- Wichtige Befehle beginnen mit `rpm -q` (`-q` = query/abfragen).
- Standardmässig liest RPM Informationen aus der lokalen Paketdatenbank der Distribution aus.

Befehl	Beschreibung
`rpm -qa`	Listet alle installierten Pakete
`rpm -qf FILENAME`	Ermittelt, welches Paket FILENAME bereitsstellt
`rpm -q PAKET`	Listet die derzeit installierte Version von Paket auf
`rpm -qi PAKET`	detaillierte Informationen zu einem Paket
`rpm -ql PAKET`	Listet die durch das Paket installierten Dateien auf
`rpm -qc PAKET`	Listet die durch das Paket installierten Konfigurationsdateien auf
`rpm -qd PAKET`	Listet die durch das Paket installierten Dokumentationsdateien auf
`rpm -q --scripts PAKET`	Listet alle Skripts auf, die beim Installieren oder Entfernen ausgeführt werden.
`rpm -q --changelog PAKET`	Zeigt das Änderungsprotokoll des Pakets an.

- falls Paket noch nicht installiert ist -> `-p` ergänzen
- `rpm -qlp DATEI` Listet alle Dateien auf, die ein lokales RPM-Paket installieren würde.
- `sudo rpm -ivh DATEI.rpm` Installiert ein lokal heruntergeladenes RPM-Paket.
  - `-i` = install
  - `-v` = ausführliche Ausgabe
  - `-h` = Hash-Anzeige während der Installation
- Kontrolle: `rpm -q DATEI`
- Paket wieder vom System entfernen: `sudo rpm -evh DATEI`
  - `-e` : erase

Beispiele

```bash
# Informationen für ein noch nicht installiertes Package anzeigen
rpm -q -p admin-scripts-1-0.noarch.rpm -i
# enthaltene Dateien für ein noch nicht installiertes Package anzeigen
rpm -q -p admin-scripts-1-0.noarch.rpm -l
# Änderungsprotokoll anzeigen
rpm -q -p admin-scripts-1-0.noarch.rpm --changelog
```

10.2.4. Extrahieren von RPM Dateien - rpm2cpio

- `rpm2cpio DATEI.rpm` Konvertiert ein RPM-Paket in ein cpio-Archiv.
- `cpio -idv` Extrahiert Dateien aus dem cpio-Archiv.
  - `-i` = extrahieren
  - `-d` = Unterverzeichnisse anlegen
  - `-v` = ausführliche Ausgabe
- Kombination zum Extrahieren aller Dateien: `rpm2cpio DATEI.rpm | cpio -idv`
- Einzelne Datei extrahieren (mit Pfadangabe): `rpm2cpio DATEI.rpm | cpio -id "*/pfad/zur/datei"`
- Dateien im RPM ohne Extrahieren anzeigen: `rpm2cpio DATEI.rpm | cpio -tv`

10.3. DNF

DNF (Dandified Yum): Paketmanagment für RPM-basierte Linux-Systeme
- Kommandozeilentool zum Suchen, Installieren und Aktualisieren von Paketen
- Plugin-Schnittstelle fuer Erweiterungen; per Python nutzbar
- Entstand 2012 als Fork von YUM 3.4
- Seit Fedora 18 verfügbar, seit Fedora 22 Standard statt YUM
- weiterhin kompatibel mit YUM

Eigenschaften:
- Dynamische Quellenauswahl während des Betriebs
- Unterstützt Python-Plugins
- Löst Paketabhängigkeiten automatisch

10.3.1. DNF vs. YUM

- DNF hat YUM als Paketmanager in RHEL 9 ersetzt
- DNF-Befehle sind funktional identisch zu YUM-Befehlen
- Aus Kompatibilitaetsgründen existieren YUM-Befehle weiterhin als symbolische Links zu DNF
- Mit `dnf` können Softwarepakete und deren Abhängigkeiten installiert, aktualisiert und entfernt werden
- `dnf` liefert zudem Informationen zu Paketen, Transaktionsverlauf und arbeitet mit mehreren Repositories

10.3.2. Befehle

Befehl	Beschreibung
`dnf help`	Informationen zu Nutzung
`dnf list` , `dnf list 'http*'`	zeigt installierte & verfügbare Pakete
`dnf search KEYWORD` , `dnf search all KEYWORD`	suchen von Paketen mit Schlüsselbegriff
`dnf info PACKAGE_NAME`	detaillierte Informationen zu einem Paket
`dnf provides PATH_NAME`	zeigt Pakete, die zu einem Pfadnamen passen (meist mit Platzhalterzeichen)
`dnf install PACKAGE_NAME`	abrufen & installieren von Sofwarepacketen inkl. Abhängigkeiten, Option `-y` → automatische Bestätigung
`dnf update PACKAGE_NAME`	abrufen & installieren der neusten Version des Pakets (inkl .Abhängikeiten)
`dnf update`	installieren aller relevanten Updates
`dnf list kernel`	Listet alle installierten Kernel-Pakete auf
`uname -r`	Laufende Kernel-Version und Release
`uname -a`	Vollstaendige System- und Kernel-Infos
`dnf remove PACKAGE_NAME`	Entfernt ein installiertes Paket inkl. Abhängigkeiten. Entfernt auch Pakete, die das Zielpaket benoetigt hat -> kann zu unerwarteten Loeschvorgaengen fuehren

10.3.3. Sofware Gruppen

Sammlung zusammengehöriger Sofware-Paketen

- Zwei Arten von Gruppen in RHEL 9:
  - Reguläre Gruppen: Paket-Sammlungen
  - Umgebungsgruppen: Sammlungen regulärer Gruppen

Befehl	Beschreibung
`dnf group list`	Zeigt verfügbare Gruppen und Umgebungsgruppen
`dnf group list hidden`	Zeigt auch die standardmässig versteckten Gruppen
`dnf group info "GRUPPE"`	Zeigt Informationen zu einer Gruppe an
`dnf group install "GRUPPE"`	installiert eine Gruppe

10.3.4. Paketmodul-Streams

Erlaubt mehrere Versionen derselben Software parallel
- RHEL 8 führt Application Streams mit Profilen ein

Version bedeutet:
- Bestimmte Versionsnummer (-> Stream)
- Bestimmter Inhalt/Ausprägung (-> Profil)
- Mehrere Streams, aber nur ein Profil

Streams
- Definieren die Versionsnummer einer Anwendung.
- Mehrere Streams können gleichzeitig verfügbar sein.
- Abhängigkeiten können in mehreren Versionen existieren.
- Hauptstreams in RHEL:
  - BaseOS
  - Anwendungsstream
- Beispiele: `python: 2.X` / `3.X`

Profile
- Bestimmen den Inhalt bzw. Umfang innerhalb eines Streams.
- Pro Stream lässt sich nur ein Profil installieren.
- Beispiele: `minimal`, `common`, `developer`

10.3.4.1. Verwalten von Streams

Befehl	Beschreibung
`dnf module list`	Zeigt alle verfügbaren Module mit Namen, Stream, Profilen und einer Kurzinfo.
`dnf module list MODULE_NAME`	Listet alle Streams eines bestimmten Moduls und deren Status.
`dnf module info MODULE_NAME`	Zeigt Details zu einem Modul (Profile, Inhalt & dazugehörige Pakete). • Ohne Streamangabe werden Standardprofil und Standardstream gezeigt. • Mit Format `modulname:stream` wird ein bestimmter Stream angezeigt. • Mit `--profile` werden die Pakete je Profil angezeigt.
`dnf module provides PACKAGE`	Zeigt, welches Modul ein bestimmtes Paket bereitstellt.

10.3.5. Transaktionen

Alle Installations- und Entfernaktionen werden in `/var/log/dnf.rpm.log` gespeichert.

Befehl	Beschreibung
`dnf history`	zeigt eine Liste aller vergangenen Transaktionen
`dnf history info ID`	zeigt Infos zu einer Transaktion
`dnf history undo ID`	macht eine bestimmte Transaktion rückgängig. entfernt, bzw. installierte Pakete automatisch
`dnf history redo ID`	Führt eine frühere Transaktion erneut aus.
`dnf history rollback ID`	Setzt das System auf den Zustand vor einer bestimmten Transaktion zurück.

10.3.6. Repo-Verwaltung

Ein Repository ist eine Paketquelle, aus der DNF Software installieren oder aktualisieren kann.

Red Hat stellt eigene offizielle Repositories bereit (z. B. BaseOS, AppStream).

Zusätzlich gibt es Drittanbieter-Repositories, die von externen Herstellern betrieben werden (z. B. Adobe, EPEL, interne Firmenserver).

Befehl	Beschreibung
`dnf repolist all`	zeigt alle verfügbaren Repositories inkl. Status an
`dnf config-manager --enable REPO`	aktiviert ein spezifisches Repo
`dnf config-manager --disable REPO`	deaktiviert ein spezifisches Repo (wird vom Packetmanager ignoriert)
`dnf config-manager --add-repo="URL"`	fügt ein neues Repository hinzu. DNF erstellt automatische eine passende `.repo`-Datei

Speicherort der Repository-Dateien

Alle Repo-Konfigurationen liegen unter: `/etc/yum.repos.d/`

Die `.repo`-Dateien enthalten u. a.:
- Name des Repos
- Basis-URL
- Status (enabled/disabled)

10.3.7. Beispiel Repo

```txt
[mirror.init7.net_federoa_epel_9_Everything_x86_64_]
name=created by dnf config-manager from https://mirror.init7.net /fedora/epel/9/
Everything/x86_64/
baseurl=https://mirror.init7.net/fedora/epel/9/Everything/x86_64/
enabled=1
```

manuell erstellt:

```txt
[epel-next]
name=epel next
baseurl=https://ftp-stud.hs-esslingen.de/pub/Mirrors/epel/next/9/ Everything/x86_64/
enabled=1
gpgcheck=0
```

`gpcheck`

- überprüft, ob das heruntergeladene Softwarepacket echt ist (nicht von Dritten manipuliert)
- `1` = Überprüfung der Autehntizität durch Überprüfugn der GPG-Signatur
- `0` = keine Überprüfung

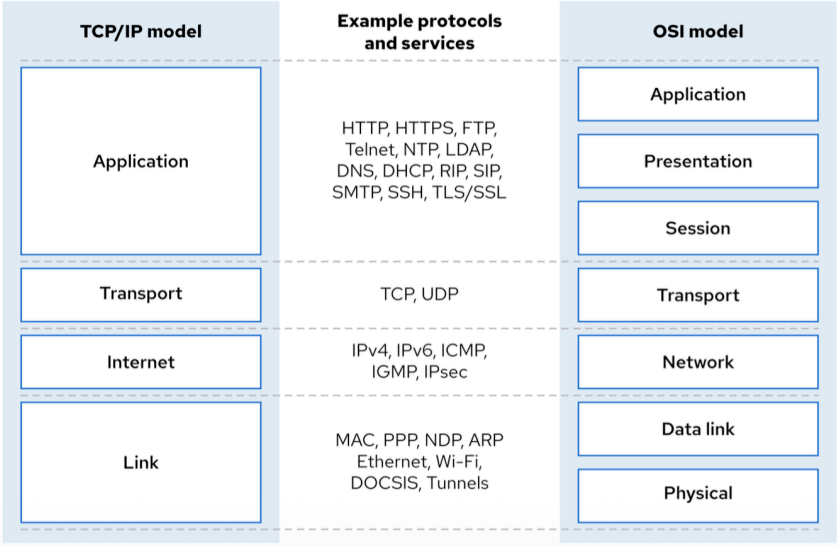
11. Netzwerk

The Ether - Zeichnungen von Robert Metcalfe, 1973 und 1976
1976: Dieses Diagramm wurde 1976 von Robert M. Metcalfe von Hand gezeichnet und von Dave R. Boggs fotografiert, um ein 35-mm-Dia zu produzieren, das verwendet wurde, um Ethernet auf der National Computer Conference im Juni desselben Jahres vorzustellen. Auf der Zeichnung sind die ursprünglichen Begriffe zur Beschreibung von Ethernet dargestellt

11.1. TCP/IP Modell

Das TCP/IP-Modell ist ein vereinfachtes 4-Schichten-Netzwerkmodell.

Es beschreibt, wie verschiedene Protokolle zusammenarbeiten, um Daten über das Internet zu übertragen.

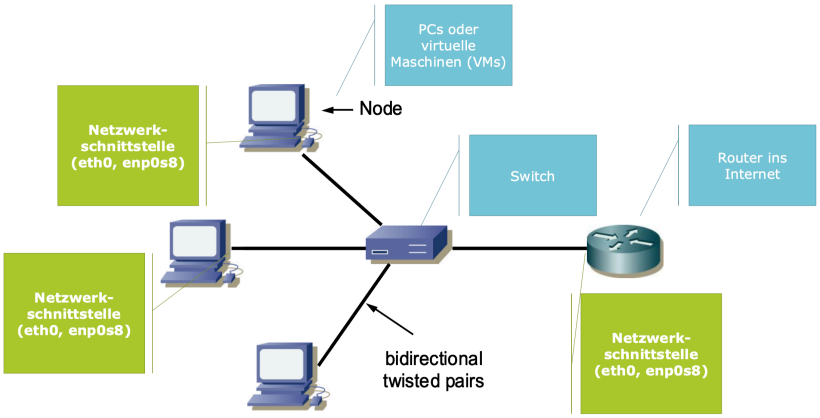


1. Application Layer
   - Kommunikation zwischen Programmen (z. B. Webserver, SSH, FTP).
   - Beispiele: HTTP, HTTPS, FTP, DNS, DHCP, SMTP, SSH, TLS.
2. Transport Layer
   - End-zu-End-Datenübertragung zwischen Anwendungen.
   - Beispiele: TCP, UDP
3. Internet Layer
   - Übertragung zwischen Netzwerken (Routing im Internet).
   - Beispiele: IPv4, IPv6, ICMP, IPsec
4. Link Layer
   - Datenübertragung im lokalen Netzwerk.
   - Beispiele: Ethernet, Wi-Fi, MAC, ARP, PPP

11.2. IPv4 vs. IPv6

- IPv4 ist aktuell das wichtigste und am weitesten genutzte Internetprotokoll.
- IPv6 wurde entwickelt, um IPv4 langfristig zu ersetzen (grösserer Adressraum, moderneres Design).
- Red Hat Enterprise Linux nutzt standardmässig Dual-Stack, d.h. IPv4 und IPv6 laufen gleichzeitig und parallel.
- IPv4: 32 bit
- IPv6: 128 bit

11.3. Aufbau eines modernen lokalen Netzwerks



11.4. IP-Adressen & Netzwerkschnittstellen

Netzwerkschnittstelle

- Verbindungspunkt zwischen PC und Netzwerk.
- Beim Router ist jede Schnittstelle einem eigenen Netzwerk zugeordnet.
- Beispiele: eth0, enp0s8

IP-Adresse

- Eindeutige Adresse zur Kommunikation im Netzwerk.
- Jeder Host besitzt mindestens eine IP-Adresse.
- Ein Router hat für jede Schnittstelle eine eigene IP-Adresse und verbindet dadurch mehrere Netzwerke.
- Netzmaske / Netzewrkpräfix bestimmt die Anzahl führenden Bits in der IP-Adresse

11.4.1. Netzwerkschnittstelle

Logischer Name einer Netzwerkverbindung unter Linux

Zugeordnet zu einer Netzwerkkarte (LAN, WLAN, WWAN)

Früher: eth0, eth1, eth2

Heute: Typbasierte Präfixe

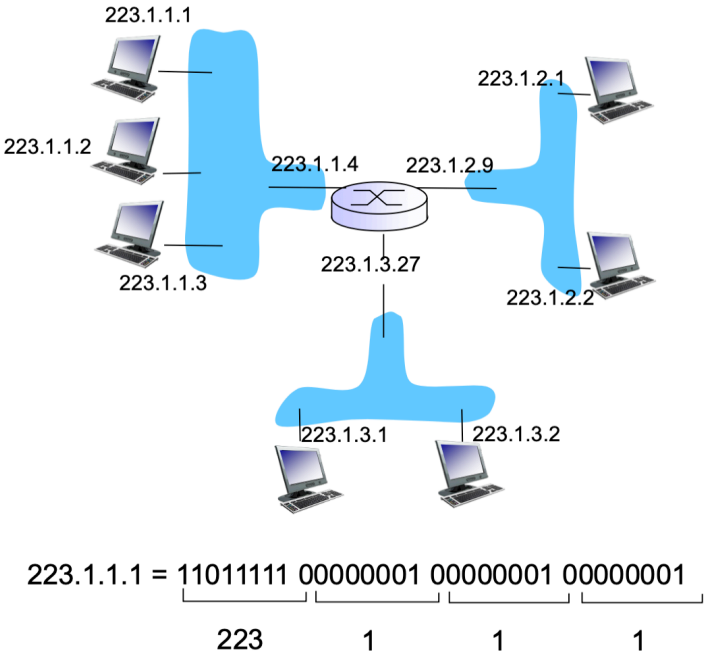
- **en** -> Ethernet
- **wl** -> WLAN
- **ww** -> WWAN

Rest des Namens abhängig von Hardwarestandort

- `o<n>` -> integriertes Gerät (z. B. eno1)
- `s<n>` -> Hotplug-PCI-Steckplatz (z. B. ens3)
- `p<m>s<n>` -> PCI-Bus M, Steckplatz N (z. B. wlp4s0)

11.4.2. IP-Adressierung

- 32-bit-Adresse für eine Netzwerkschnittstelle
- Router: mehrere IP-Adressen (eine pro Schnittstelle)
- PCs: meist 1–2 IP-Adressen (LAN, WLAN)
- Jede IP-Adresse gehört genau zu einer Schnittstelle
- IP-Adresse bestimmt das Netzwerk, in dem das Gerät liegt



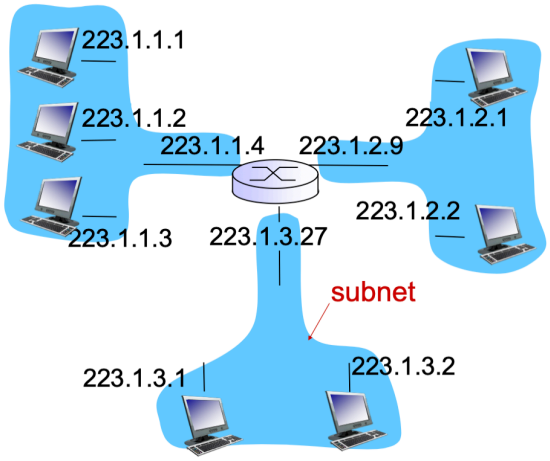
11.5. Subnetze

IP-Adressen bestehen aus:

- Netzwerk-Adressbereich / Subnetz (höherwertige Bits)
- Host-Adressbereich (niederwertige Bits)

Subnetz = Gruppe von Interfaces mit dem gleichen Netzwerkanteil

- Geräte im gleichen Subnetz: direkt erreichbar, ohne Router
- Netzwerk aus mehreren Subnetzen → Router verbindet sie

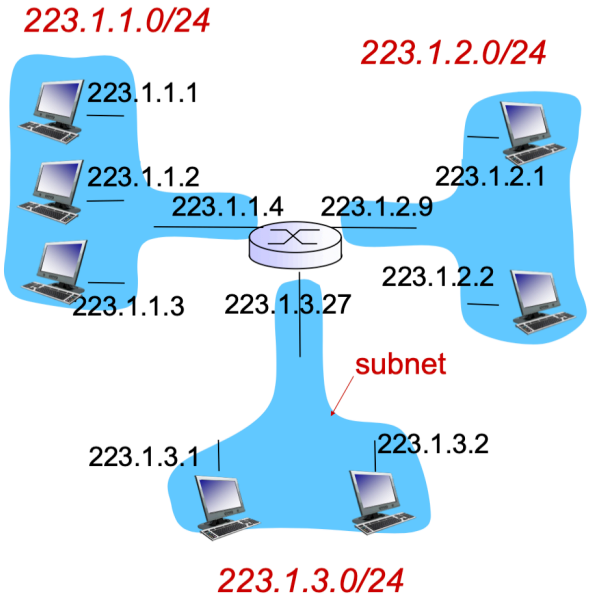


Netzwerk bestehend aus 3 Subnetzen

11.5.1. Identifizierung von Subnetzen

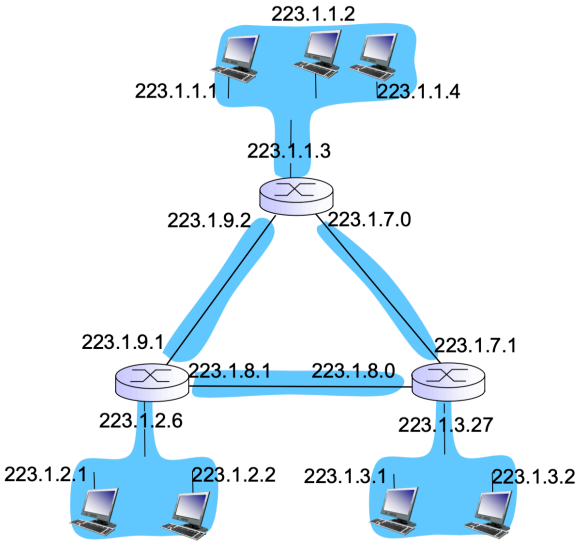
Rezept zur Identifizierung

- Betrachte jedes Interface losgelöst von seinem Host oder Router
- Bilde Inseln von isolierten Netzwerken
- Jedes isolierte Netzwerk ist ein Subnetz



subnet mask: /24

11.5.2. Subnetze mit mehreren Routern



11.6. IP-Adressierung - CIDR

CIDR = Classless Inter-Domain Routing

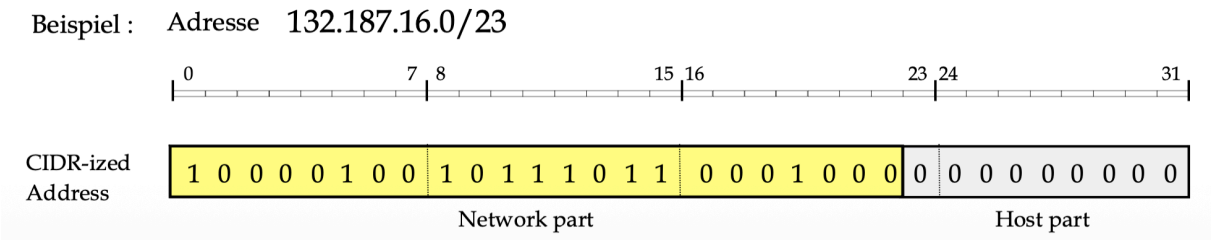
IP-Adresse besteht aus: Netzwerkpräfix + Host-Teil

Länge des Netzwerkanteils ist variabel

Adressformat: a.b.c.d/x

x = Anzahl Bits des Netzwerkpräfix

Beispiel: 132.187.16.0/23 ⟶ 23 Bits = Netzwerk, Rest = Hosts



11.6.1. Gültige IPv4 Adressen

- /8 = Maske 255.0.0.0 (1 Block fest)
- /16 = Maske 255.255.0.0 (2 Blöcke fest)
- /24 = Maske 255.255.255.0 (3 Blöcke fest)
- .0 ist reserviert für das Netzwerk selbst
- .255 ist für den Broadcast reserviert
- gültig zwischen 1 und 254

11.6.1.1. Beispiele

Netzwerk-Typ	Beispiel-Netz	Feste Blöcke	Erste Host-IP	Letzte Host-IP	Broadcast
/8	10.0.0.0/8	10.	10.0.0.1	10.255.255.254	10.255.255.255
/16	172.16.0.0/16	172.16.	172.16.0.1	172.16.255.254	172.16.255.255
/24	192.168.1.0/24	192.168.1.	192.168.1.1	192.168.1.254	192.168.1.255

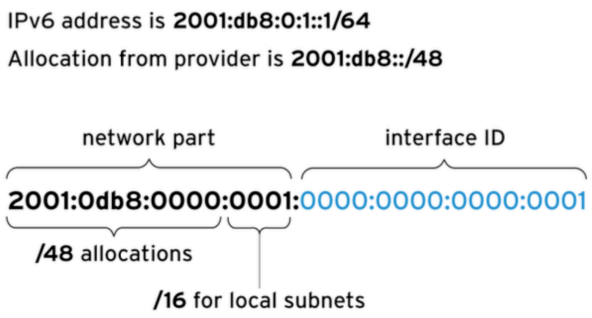
11.7. Routing im Netzwerk

- Daten müssen zwischen Hosts und zwischen Netzwerken übertragen werden
- **Verbindung zwischen Netzen = Routing**
- Jedes System entscheidet selbst, welche Netze über welche Schnittstelle erreichbar sind

Ziel	Schnittstelle	Router falls nötig
`192.168.2.0/24`	wlo1	
`192.168.5.0/24`	enp0s3	
`0.0.0.0 - default`	enp0s3	192.168.2.1

11.8. IPv6 Adressierung

- IPv6-Adresse besteht aus Netzwerkpräfix und Schnittstellen-ID
- Netzwerkpräfix -> identifiziert das Subnetz
- Schnittstellen-ID ⟶ eindeutige Interface-Adresse innerhalb des Subnetzes
- In einem Subnetz dürfen keine zwei Interfaces dieselbe ID haben
- Standard-Subnetzmaske bei IPv6: /64
- Bedeutet:
  - erste 64 Bit = Netzwerk
  - letzte 64 Bit = Interface ID
- Provider vergeben oft grössere Präfixe, z. B. /48



11.9. Netzwerkkonfiguration

Befehl	Beschreibung
`ip link`	Zeigt alle Netzwerkinterfaces und deren Status
`ip addr`	Listet IP-Adressen aller Interfaces
`ip addr show dev eth0`	IP-Infos für eth0 anzeigen
`ip addr show dev enp0s3`	IP-Infos für enp0s3 anzeigen
`ip addr show dev eth1`	Interface hat keine IPv4/IPv6-Adresse
`nmcli dev status`	Zeigt allgemeine Device-Übersicht
`nmcli dev show eth1`	Details zum Interface eth1
`nmcli con show`	Listet alle Verbindungen
`nmcli con show --active`	Listet aktive Verbindungen
`nmcli con show "System eth1"`	Details zur Verbindung „System eth1“
`vim /etc/NetworkManager/system-connections/ System eth1.nmconnection`	Manuelle Anpassung der Konfigurationsdatei
`ip -s link show enp0s3`	Statistiken zu einem Interface
`ip route`	Zeigt die Routing-Tabelle
`sudo ip route add 4.2.2.4 via 172.30.1.254`	Temporäres Routing (iproute2)
`ping -c3 8.8.8.8`	Sendet drei ICMP-Pakete zum Test
`tracepath access.redhat.com`	Zeigt den Paketpfad zum Ziel
`ss -ta`	Anzeige aktiver TCP-Verbindungen
`ss -tl`	Liste lauschender TCP-Ports
`ss -ua`	Anzeige aktiver UDP-Verbindungen
`nmcli con add con-name enp0s4-con type ethernet ifname eth1 ipv4.address 192.168.0.5/24 ipv4.gateway 192.168.0.254`	Neue Verbindung für eth1 mit IP, Netzmaske und Gateway
`nmcli con mod "enp0s4-con" ipv4.dns 4.2.2.4`	DNS für die Verbindung setzen
`nmcli con mod "enp0s4-con" connection.autoconnect no`	Autoconnect deaktivieren
`nmcli dev status`	Status erneut pruefen
`nmcli dev show eth1`	Details zum Device nach Änderungen
`nmcli con show`	Alle Verbindungen anzeigen
`nmcli con show --active`	Aktive Verbindungen anzeigen
`nmcli con up "enp0s4-con"`	Verbindung aktivieren
`sudo nmcli con mod static-addr +ipv4.routes "4.2.2.4 172.30.1.254"`	Persistentes Routing (nmcli)

`ip addr show dev xxx` : falls `inet` bzw. `inet6` Einträge fehlen, ist die IP-Schicht nicht aktiv

11.9.1. Möglichkeiten für die Netzwerkonfiguration

1. `iproute2` -Kommando, Befehl `ip ...`
   - für Netzwerkeinstellungen
   - Änderungen gehen nach einem Neustart verloren
2. das Programm `NetworkManager`, Befehl `nmcli ...` (empfohlen)
   - RedHat eigene Tool
3. GNOME control-center (via GNOME Desktop-Umgebung)
4. Konfigurationsdatei `/etc/NetworkManager/system-connections/< name>.nmconnection`
   - Manuelle Anpassung
   - nach Anpassung Verbindungen mit `nmcli con reload` neu einlesen

11.9.2. Ändern der Netzwerkkonfiguration

`/etc/NetworkManager/system-connections/`

- Persistente Profile, die von Benutzern erstellt oder angepasst wurden
- NetworkManager synchronisiert diese automatisch in dieses Verzeichnis.

`/run/NetworkManager/system-connections/`

- Temporäre Profile, die nur bis zum nächsten Neustart bestehen.

`/usr/lib/NetworkManager/system-connections/`

- Vom System oder durch Deployment bereitgestellte, unveränderliche Standardprofile.
- Werden diese bearbeitet, kopiert NetworkManager sie automatisch nach persistent oder temporär, je nach Einsatz.

Nach Anpassungen:

1. Berechtigungen korrekt setzen (Nur root darf lesen und schreiben)
   - `chown root:root /etc/NetworkManager/system-connections/"Main eth0.nmconnection"`
   - `chmod 600 /etc/NetworkManager/system-connections/"Main eth0.nmconnection"`
2. Änderungen einlesen
   - `nmcli con reload`
   - `nmcli con up "static-ens3"`
3. Wenn autoconnect deaktiviert ist, muss Verbindung manuell gestartet werden
   - `nmcli con up <name>`

Beispiel einer IPv4-Konfigurationsdatei (statisch):

```
[connection]
id=Main eth0
uuid=27afa607-ee36-43f0-b8c3-9d245cdc4bb3
type=802-3-ethernet
autoconnect=true

[ipv4]
method=auto

[802-3-ethernet]
mac-address=00:23:5a:47:1f:71
```

11.9.3. `nmcli con mod` Optionen

nmcli	Datei `*.nmconnection`	Auswirkung
`ipv4.method manual`	`[ipv4]` `method=manual`	Konfiguriert IPv4-Adressen statisch
`ipv4.method auto`	`[ipv4]` `method=auto`	Sucht nach Einstellungen von einem DHCPv4-Server
`ipv4.addresses 192.0.2.1/24`	`[ipv4]` `address1=192.0.2.1/24`	Definiert statische IPv4-Adresse und Praefix
`ipv4.gateway 192.0.2.254`	`[ipv4]` `gateway=192.0.2.254`	Setzt das Standard-Gateway
`ipv4.dns 8.8.8.8`	`[ipv4]` `dns=8.8.8.8`	Schreibt Nameserver in **resolv.conf**
`ipv4.dns-search example.com`	`[ipv4]` `dns-search=example.com`	Setzt Suchdomain in **resolv.conf**
`ipv4.ignore-auto-dns true`	`[ipv4]` `ignore-auto-dns=true`	Ignoriert DNS-Infos vom DHCP-Server
`ipv6.method manual`	`[ipv6]` `method=manual`	Konfiguriert IPv6-Adressen statisch
`ipv6.method auto`	`[ipv6]` `method=auto`	SLAAC-Konfiguration ueber Router-Advertisements
`ipv6.method dhcp`	`[ipv6]` `method=dhcp`	DHCPv6, kein SLAAC
`ipv6.addresses 2001:db8::a/64`	`[ipv6]` `address1=2001:db8::a/64`	Setzt statische IPv6-Adresse und Praefix
`ipv6.gateway 2001:db8::1`	`[ipv6]` `gateway=2001:db8::1`	Setzt Standard-Gateway
`ipv6.dns fde2:6494:1e09:2::d`	`[ipv6]` `dns=fde2:6494:1e09:2::d`	Schreibt IPv6-Nameserver in **resolv.conf**

nmcli	Datei `*.nmconnection`	Auswirkung
`ipv6.dns-search example.com`	`[ipv6]` `dns-search=example.com`	Setzt IPv6-Suchdomain
`ipv6.ignore-auto-dns true`	`[ipv6]` `ignore-auto-dns=true`	Ignoriert DHCPv6-DNS-Infos
`connection.autoconnect yes`	`[connection]` `autoconnect=true`	Aktiviert Verbindung automatisch beim Booten
`connection.id ens3`	`[connection]` `id=Main eth0`	Name der Verbindung
`connection.interface-name ens3`	`[connection]` `interface-name=ens3`	Bindet Verbindung an Interface-Namen
`802-3-ethernet.mac-address …`	`[802-3-ethernet]` `mac-address=…`	Bindet Verbindung an bestimmte MAC-Adresse

12. Skripte & Automatisierung

12.1. Bash-Skripte

DEFINITION: Ein Bash-Skript ist eine ausführbare Datei mit Befehlen und einfacher Programmierlogik

- Administration in Linux erfolgt meist über Kommandozeilen-Tools
- Komplexere Aufgaben brauchen oft eine Verkettung mehrerer Befehle
- Wiederkehrende oder schwierige Aufgaben werden per Bash-Skripts automatisiert
- Shell-Scripting ist zentral für erfolgreiche Systemadministration
- Erstellung: leere Datei im Editor anlegen und Befehle einfügen
- Editoren wie vim/emacs bieten Syntax-Highlighting für Bash
- Syntaxfehler wie fehlende Anführungszeichen oder Klammern lassen sich damit leichter vermeiden

12.2. Festlegen des Befehlsinterpreters - Shebang

- Erste Zeile im Skript gibt den Interpreter an
- Format: `#!/usr/bin/env bash` (oder zsh, python usw.)
- Wird Shebang / Hash-Bang genannt
- Startet immer mit `#! + Pfad` zum Interpreter

12.2.1. `#!/usr/bin/env bash` vs. `#!/bin/bash`

`#!/bin/bash`

- direkter Pfad
- Vorteil: eindeutig
- Nachteil: Wenn Bash auf einem System an einem anderen Ort installiert ist (z. B. unter /usr/local/bin/bash bei BSD oder macOS), schlägt das Skript fehl.

`#!/usr/bin/env bash`

- nutzt Hilfsprogramm `env`
- sucht in Umgebungsvariable `$Path` des Benutzers nach dem ersten Vorkommen von bash
- Vorteil: wird gefunden, solange im Pfad des Nutzers vorhanden
- Nachteil: minimales Sicherheitsrisiko, da ein Nutzer eine „gefälschte“ Bash-Version weiter vorne in seinem Pfad platzieren könnte.

12.3. Ausführen eines Skripts

- Shell-Skripte wie normale Befehle ausführbar
- Ausführung direkt: `bash script.sh`
- Oder Datei selbst ausführbar machen: `chmod u+x script.sh`
- Danach: `./script.sh`

12.4. Speicherort & `$PATH`

- Skript in ein Verzeichnis legen, das im `$PATH` steht
- Dann ist Aufruf ohne Pfadangabe möglich → `hello-world.sh` statt `./hello-world.sh`
- Interpreter nimmt die erste Datei aus `$PATH`, die den Namen trägt
- Absolute oder relative Pfade funktionieren weiterhin
- Keine Skriptnamen verwenden, die gleich heissen wie bestehende Befehle (z.B. ls, echo)!

12.5. Ausgaben

`echo`

- Gibt Text aus, der als Argument an die Shell übergeben wird
- Standardausgabe: STDOUT
- Kann auch auf STDERR umgeleitet werden (>&2)

Verwendung in Shell-Skripten

- `echo` sehr häufig für Status-, Info- und Fehlermeldungen
- Fortschrittsanzeigen in Skripten:
  - an STDOUT, STDERR oder in Logdateien umleitbar
- Zweck von Umleitungen:
  - Trennung von normalen Meldungen (STDOUT) und Fehlern (STDERR)

Beispiel:

- STDOUT auf Konsole
- STDERR in Log-File

```sh
[user@host ~]$ cat ~/bin/hello
#!/bin/bash

echo "Hello, world"
echo "ERROR: Houston, we have a problem." >&2


[user@host ~]$ hello 2> hello.log
Hello, world
[user@host ~]$ cat hello.log
ERROR: Houston, we have a problem.
```

12.5.1. Umgang Sonderzeichen `\` / `' '` / `" "`

- Manche Zeichen haben in Bash eine besondere Bedeutung
- Manchmal müssen Zeichen wörtlich ausgegeben werden
- Drei Methoden zum Maskieren:
  - Backslash `\`
  - Einfache Anführungszeichen `' '`
  - Doppelte Anführungszeichen `" "`

Backslash-Escape `\`

- Entfernt Sonderbedeutung eines einzelnen Zeichens
- Beispiel: `\#` verhindert Kommentarinterpretation
- Nur das nächste Zeichen wird maskiert

```sh
[user@host ~]$ echo # not a comment

[user@host ~]$ echo \# not a comment
# not a comment
```

Einfache Anführungszeichen `' '`

- Interpretiert alles wörtlich
- Kein Globbing, keine Variablenersetzung, keine Befehlsersetzung
- Meta-Zeichen wie `?` bleiben unverarbeitet
- Ideal für Texte, die exakt ausgegeben werden sollen

Doppelte Anführungszeichen `" "`

- Unterdrücken Globbing und Shell-Expansion
- Variablen- und Befehlsersetzung bleiben aber erlaubt
- Beispiel: `"Hostname ist ${var}"` funktioniert
- Gut für Texte, die sowohl wörtliche Inhalte als auch Variablen enthalten

```sh
[user@host ~]$ echo "Will variable $var evaluate to $(hostname -s)?"
Will variable host evaluate to host?

[user@host ~]$ echo 'Will variable $var evaluate to $(hostname -s)?'
Will variable $var evaluate to $(hostname -s)?
```

12.6. Variablen

DEFINITION: Information im Speicher zu speichern um später wiederverwendet werden zu können.

```bash
#!/usr/bin/env bash

#
# Author : Rocky Documentation Team
# Date : March 2022
# Version 1.0.0: Save in /root the files passwd, shadow, group, and gshadow

# Global variables
FILE1=/etc/passwd
FILE2=/etc/shadow
FILE3=/etc/group
FILE4=/etc/gshadow

# Destination folder
DESTINATION=/root

# Clear the screen
clear

# Launch the backup
echo "Starting the backup of $FILE1, $FILE2, $FILE3, $FILE4 to $DESTINATION:"

cp $FILE1 $FILE2 $FILE3 $FILE4 $DESTINATION

echo "Backup ended!"
```

- **Inhalt**: Zeichen oder Zeichenfolge
- **Erstellung**: wenn Variable ihren Inhalt erhält
- **Gültigkeit**: bis Ende der Ausführung des Skripts oder wie festgelegt
- **Aufrufbarkeit**: Eine Variable kann auch vor ihrer Erstellung aufgerufen werden
- Inhalt einer Variable kann sich während des Skripts ändern
- bei Löschung der Variable während des Skripts, bleibt diese bestehen ist aber leer
- Variablentyp ist möglich, wird aber in einem Shell-Skript selten verwendet
- Ergebnis eines Befehls kann in einer Variablen gespeichert werden

```bash
#!/bin/bash

# Ruft den Hostnamen (kurz) ab und speichert ihn in der Variablen 'var'
var="$(hostname -s)"

# Gibt den Inhalt der Variablen aus
echo "Name des Hosts: $var"
```

12.7. Umgebungsvariablen / Systemvariablen

- Variablen, die das System für seinen Betrieb verwendet
- Benennung in Grossbuchstaben
- können bei der Ausführung eines Skripts angezeigt werden
- können geändert werden (nicht empfohlen)
- `env` : zeigt alle verwendeten **Umgebungsvariablen** an
- `set` : zeigt alle verwendeten **Systemvariablen** an

12.7.1. Umgebungsvariablen für Shell-Skripte

Variable	Wirkweise
`$HOSTNAME`	Hostname der Maschine.
`$USER, $USERNAME, $LOGNAME`	Name des Benutzers, der mit der Sitzung verbunden ist.
`$PATH`	Liste an Verzeichnissen, die der Interpreter durchsucht, wenn **kein** Pfad vor dem Befehl beim Ausführen eines Befehls angegeben wird.
`$PWD`	Aktuelles Verzeichnis, jedes Mal aktualisiert, wenn der Befehl `cd` ausgeführt wird.
`$HOME`	Login-Verzeichnis.
`$$`	Prozess-ID der Skriptausführung.
`$?`	Rückkehrcode des zuletzt ausgeführten Befehls.

12.8. Variablen belegen

WICHTIG: **keine** Leerzeichen vor und nach `=` verwenden
Bsp. `message=hallo`

Zuweisung enthält Leer- oder Sonderzeichen → Anführungszeichen
Bsp. `message="hallo Welt"`

oder maskieren mittels Backslashs (für Leerzeichen / Sonderzeichen)
Bsp. `message=hello\ world`

12.9. Variablen lesen und löschen

Variablen ausgeben: `echo` (vor der Variable muss ein `$` und „ stehen)
Bsp. `echo "$message"`

Variablen löschen: `unset`
Bsp. `unset message`

12.10. Variablen in Zeichenketten verwenden

geschweifte Klammer für die Variable verwenden

Bps. `echo "${message}lolo"`

Ausgabe: Hallololo

ohne geschweifte Klammern gibt es einen Fehler `echo "$messagelolo"`

12.11. Ausgaben in eine Variable schreiben

Befehlssubstitution = Textausgabe eines Befehls in Variable speichern

Beispiel: Liste der Man-Pages im System mit dem Stichwort „player“ anzeigen

```bash
#!/bin/bash

# Ausgaben in Variable schreiben

# Variablendefinition
suchwort=player
liste="$(apropos $suchwort)"

echo "Player-Liste:"
echo "$liste"
```

Alles zwischen `$(` und `)` wird als Befehl ausgeführt und die Ausgabe in die Variable `$liste` gespeichert.

-> Daten können mit `echo` ausgegeben werden (oder Schleife)

nicht empfohlene alternative Schreibweise: Backticks

Bps.

```bash
`: NAME=`whoami`.
```

12.12. Schleifen

12.12.1. if/then-Konstrukts

```bash
if <CONDITION>; THEN
  <STATEMENT>
  ...
  <sTATEMENT>
fi
```

- wenn Bedienung erfüllt, werden ein oder mehrere Aktionen ausgeführt
- keine Ausführung wenn Bedingung nicht erfüllt

12.12.2. if/then/elif/then/else-Konstrukts

Beispiel:

- mysql-Client ausführen, wenn mariadb-Dienst aktiv ist, den psql-Client ausführen, wenn der postgresql-Dienst aktiv ist oder den sqlite3-Client ausführen, wenn beider der Fall ist
- mariadb- und postgresql sind beide nicht aktiv
- 0 = erfolgreich (hier Dienst ist aktiv)

```bash
systemctl is-active mariadb > /dev/null 2>&1
MARIADB_ACTIVE=$?

systemctl is-active postgresql > /dev/null 2>&1
POSTGRESQL_ACTIVE=$?

if [ "$MARIADB_ACTIVE" -eq 0 ]; then
    mysql
elif [ "$POSTGRESQL_ACTIVE" -eq 0 ]; then
    psql
else
    sqlite3
fi
```

12.12.3. Vergleich: Ausgabe und Rückgabe bei Linux-Prozessen `STDOUT` und `STDERR`, Prozessrückgabewert

- `STDOUT` und `STDERR`
  - **Ausgabe Streams**
  - Prozess kann Hinweise, Informationen und Fehler ausgeben
- **Prozess-Rückgabewert** (Exit-Code oder Exit-Status)
  - wenn Prozess beendet wird gibt er einen Exit-Status an seinen aufrufenden Prozess zurück
    - Mutterprozess kann kontrollieren, ob die Ausführung des Tochterprozesses ohne Fehler oder mit Fehlern beendet wurde
    - ähnlich zu einem return Statement
  - Abfrage mit `> echo $?`

`STDOUT` und `STDERR` vs. Prozessrückgabewerte

- sind getrennte Konzepte
- Ausgabe Streams –> geben Meldungen aus
- Prozessrückgabewerte –> zeigen erfolgreiche Ausführung

```bash
# Skript: check_file.sh
if [ -f "rechnung.pdf" ]; then
    echo "Datei gefunden!"  # <--- DAS IST DIE AUSGABE
    exit 0                  # <--- DAS IST DER RÜCKGABEWERT (Erfolg)
else
    echo "Fehler: Datei fehlt." # <--- DAS IST DIE AUSGABE
    exit 1                      # <--- DAS IST DER RÜCKGABEWERT (Fehler)
fi
```

12.12.3.1. Rückgabewerte in einem Programm oder Skript

- Programm oder Skript kann explizit mit einem Rückgabewert beendet werden
- `exit -1` → Rückgabewert −1
- `exit 0` → Rückgabewert 0
- bei erfolgreicher Beendigung gibt man 0 zurück, bei Fehlern −1, 1 oder 2 → je nach Programmierung

12.12.4. Prozess-Rückgabewert in bedingten Anweisungen

Bei einer `if`-Anweisung in der Shell (z. B. `if ./mein_skript.sh; then ...`) wird nicht die Ausgabe des Skripts (der Text, den es druckt) geprüft, sondern direkt der **Rückgabewert** des Befehls, der nach dem `if` steht.

- `hello.sh` gibt als Prozessrückgabewert `"0"` zurück

```bash
[labadmin@lios-workstation-01 tmp]$ ./hello.sh
[labadmin@lios-workstation-01 tmp]$ echo $?
0
[labadmin@lios-workstation-01 tmp] if ./hello.sh; then
echo "EXIT 0";
fi

EXIT 0
```

- `./error.sh` gibt als Prozessrückgabewert `"254"` zurück

```bash
[labadmin@lios-workstation-01 tmp] ./error.sh
[labadmin@lios-workstation-01 tmp] echo $?
254
[labadmin@lios-workstation-01 tmp] if ./error.sh; then
echo "EXIT 0";
else
echo "EXIT 254";
fi

EXIT 254
```

12.12.5. Nutzung von `test` in bedingten Anweisungen

- `test` setzt den Rückgabewert je nach angegebener Bedingung
- prüft ob angegebene Bedingung wahr sind und gibt einen Prozessrückgabewert zurück, der in bedingten Anweisungen oder Skripten zur Steuerung des Programmflusses verwendet werden kann

Beispiel: `test 1 –eq 1` Prozessrückgabewert: `0`

Beispiel: `test "asd" = "asd"` Prozessrückgabewert: `0`

Beispiel: `test "asd" != "asd"` Prozessrückgabewert: `1`

12.12.5.1. Verkürzter Syntax mit `[..]`

```bash
if [ "$(./info.sh)" = "INFO: Das ist eine Ausgabetext" ]; then
    # ...
fi
```

Die Ausgabe von `info.sh` wird mit `= "INFO: ..."` verglichen

Wichtig: Es **muss** ein Leerzeichen nach der öffnenden Klammer `[` und vor der schliessenden Klammer `]` stehen.

`if ["$A"="$B"]` (ohne Leerzeichen) würde fehlschlagen, weil die Shell nach einem Programm sucht, das `["$A"="$B"]` heisst (was es nicht gibt)

12.12.5.2. Verwendung des Ausgabestreams eines Prozesses in bedingten Anweisungen über `test`

```bash
if test "$VAR" = "INFO: Das ist eine Ausgabetext"; then
    echo "Die Ausgabe ist: INFO: Das ist eine Ausgabetext"
fi
```

- Variable speichert Inhalt des Ausgabestreams
- `test` führt Vergleich durch und gibt entsprechenden Rückgabewert zurück

12.13. Verkettung von Befehlen

12.13.1. Methoden

Methoden zur Verkettung in Bash sind folgende:

`;` → Ausführung der Befehle nacheinander ohne Bedingung

Beispiel: `ls /home/user ; ls /home/user/Desktop`

Zuerst wird das Verzeichnis /home/user aufgelistet, dann /home/iser/Desktop

---

`&&` → Zweiter Befehl wird nur ausgeführt wenn der Erste erfolgreich war

Beispiel: `mkdir newFolder && cd newFolder`

Ein Verzeichnis wird erstellt, und falls das erfolgreich war wird in das Verzeichnis gewechselt.

`||` → Zweiter Befehl wird nur ausgeführt wenn der Erste fehlerhaft war.

Beispiel: `ls /home/user/nonexistent_directory || echo "Directory not found"`

Versucht das Verzeichnis nonexisten_directory aufzulisten. Wenn das fehlschlägt wird „Directory not found“ ausgegeben.

`&` → Erste Befehl wird im Hintegrund ausgeführt, alle anderen im Vordergrund

Beispiel: `long_running_command & echo "This is done in the foreground"`

Führt den long_running_command im Hintergrund aus und gibt dann „This is done in the foreground“ aus.

`|` → Pipeline, Verkettung von Befehlen, Output des ersten Befehl ist der Input des zweite Befehls.

12.13.2. Prioritäten

- `&&` und `||` haben eine höhere Priorität als `;` und `&`
- `&&` und `||` haben die gleiche Priorität
- `;` und `&` haben die gleiche Priorität

12.13.3. Klammern

Mit Klammern kann die Priorität der Operationen überschrieben werden.

Folgende Klammern werden unterstützt:
- (Befehl1; Befehl2 && …): zur Ausführung der Befehle wird eine eigenen untergeordnete Shell gestartet
- {Befehl1; Befehl2; …;}: zur Ausführung der Befehle wird **keine** eigene Shell gestartet

Bei geschweiften Klammern muss zwingen ein abschliessendes Semikolon gesetzt werden

Beispiel: `Befehl1 && {Befehl2; Befehl3;}` → Ist Befehl1 erfolgreich, führe auch Befehl2 gefolgt von Befehl 3 aus, andernfalls weder noch.

12.14. Reguläre Ausdrücke

Mit regulären Ausdrücken kann ein Musterabgleich gemacht werden

- vim, grep und less unterstützen alle Regex
- grep → global regular expression print
- Regex ist eine eigene Sprache und hat eigene Syntax und Regeln

12.14.1. Zeilenanfang und Ende

Das Pattern `cat` sei gegeben:

- `cat` match mit jeder Zeile wo irgendwo cat drinsteht
  - Z.B, **cat**, con**cat**enate, **cat**egory, tom**cat**
- `^cat` matcht nur wenn cat am Anfang der Zeile steht
  - Z.B **cat**, **cat**egory
- `cat$` matcht nur wenn cat am Ende der Zeile steht
  - Z.B **cat**, tom**cat**
- `^cat$` matcht nur wenn cat das einzige Wort auf der Zeile ist
  - Z.B **cat**

12.14.2. Platzhalter

- `.` stimmt mit jedem Zeichen (ausser Zeilenumbruch) überein
  - Bsp. `c.t` → cat, concatenante, vindication, c5t, `c$t`

12.14.3. Multiplikatoren

- Multiplikatoren gelten für das vorherige Zeichen im regulären Ausdruck
- sie können nicht nur für Zeichen verwendet werden sondern auch für Ausdrücke
- `*` bedeutet das der vorherige Ausdruck 0 oder mehrere Male vorkommt
- Es kann auch eplizit angegeben werden wie oft der vorherige Ausdruck vorkommt mit `{n}`

Beispiele:

- `c[aou]*t` matcht zum Beispiel cut, coat, cat usw.
- `c.*t` matcht zum Beispiel cat, cut, ct, culvert, usw.
- `c.{2}t` matcht zum Beispiel cost, cast, coat, cart, usw.

`.` → Jedes Zeichen ausser ein Zeilenumbruch

`[aou]` → Matcht entweder a,o oder u

12.14.4. Übersicht

Option	Beschreibung
`.`	Matcht alle Zeichen (ausser Zeilenumbrüche)
`?`	Vorheriges Zeichen ist optional und wird maximal einmal gematcht
`*`	Vorheriges Zeichen wird 0 oder mehrere Male gematcht
`+`	Vorheriges Zeichen wird einmal oder mehrmals gematcht
`{n}`	Vorheriges Zeichen wird genau n Mal gematcht
`{n,}`	Vorherigeges Zeichen wird mindestens n Mal gematcht
`{,m}`	Vorheriges Zeichen wird höchstens m Mal gematcht
`{n,m}`	Vorheriges Zeichen wird zwischen n und m Mal gematcht
`[:alnum:]`	Matcht alle alphanumerischen Zeichen
`[:alpha:]`	Matcht alle alphabetischen Zeichen
`[:blank:]`	Matcht Spaces und Tabs
`[:cntrl:]`	Ascii Characters 000 bis 037 und 177(DEL)
`[:digit:]`	Matcht alle Ziffern
`[:graph:]`	Matcht [:alnum:] und [:punct:]
`[:lower:]`	Matcht kleingeschrieben Buchstaben
`[:print:]`	Matcht [:alnum:] und [:punct:] und space
`[:punct:]`	Matcht Satzzeichen (#$%&'()*+,-./:;<=>?@[\]^_{|})~)
`[:space:]`	Matcht, tab, newline, vertical tab, form feed, carriage return, und space
`[:upper:]`	Matcht Grossbuchstaben
`[:xdigit:]`	Matcht Hexadezimel Zeichen
`\b`	Matcht leere Strings am Rand von einem Wort
`\B`	Matcht leere Strings **nicht** Rand von einem Wort
`\<`	Matcht leere Strings am Anfang von einem Wort
`\>`	Matcht leere Strings am Ende von einem Wort
`\w`	Matcht [:alnum:]
`\W`	Matcht alles ausser [:alnum:]
`\s`	Matcht [:Space:]

Beispiel:

```
[root@lios-server-01 ~]# cat /etc/ethertypes
#
# Ethernet frame types
#       This file describes some of the various Ethernet
#       protocol types that are used on Ethernet networks.
#
# This list could be found on:
#         http://www.iana.org/assignments/ethernet-numbers
#         http://www.iana.org/assignments/ieee-802-numbers
#
# <name>    <hexnumber> <alias1>...<alias35> #Comment
#
IPv4        0800    ip ip4      # Internet IP (IPv4)
X25     0805
ARP     0806    ether-arp   #
FR_ARP      0808            # Frame Relay ARP       [RFC1701]
BPQ     08FF            # G8BPQ AX.25 Ethernet Packet
DEC     6000            # DEC Assigned proto
DNA_DL      6001            # DEC DNA Dump/Load
DNA_RC      6002            # DEC DNA Remote Console
DNA_RT      6003            # DEC DNA Routing
LAT     6004            # DEC LAT
        ...
```

```
[root@lios-server-01 ~]# grep -v '^[#;]' /etc/ethertypes
IPv4        0800    ip ip4      # Internet IP (IPv4)
X25     0805
ARP     0806    ether-arp   #
FR_ARP      0808            # Frame Relay ARP       [RFC1701]
BPQ     08FF            # G8BPQ AX.25 Ethernet Packet
DEC     6000            # DEC Assigned proto
DNA_DL      6001            # DEC DNA Dump/Load
DNA_RC      6002            # DEC DNA Remote Console
DNA_RT      6003            # DEC DNA Routing
LAT     6004            # DEC LAT
        ...
```

12.15. `grep`

DEFINITION: grep (Global Regular Expression Print) ist ein Werkzeug zum Durchsuchen von Dateien oder Datenströmen nach bestimmten Textmustern.

grep ist das zentrale Tool, um Systemprotokolle (Logs) und Konfigurationsdateien effizient zu analysieren.

Es unterstützt reguläre Ausdrücke (Regex), um komplexe Suchmuster zu definieren

12.15.1. Wichtige Optionen

Option	Beschreibung
`-i`	Ignoriert Gross- und Kleinschreibung (ignore case).
`-e`	Ermöglicht die Suche nach mehreren Mustern gleichzeitig (extended/multiple patterns). `grep -e "Fehler" -e "Warnung" system.log`
`-v`	Invertiert die Suche: Zeigt nur Zeilen an, die nicht auf das Muster passen.
`-r`	Durchsucht Verzeichnisse rekursiv.

12.15.2. Beispiele

```bash
grep '^SELINUX' /etc/selinux/config #Durchsucht config Datei nach SELINUX
```

12.16. Zukünftige und wiederkehrende Jobs (at/atq und crontab)

12.16.1. at

Neuer Job `Befehl at <ZEITANGABE>`

Nur mit dem `at` Befehl kann man eine Eingabe öffnen um den Befehl direkt einzugeben. Diese kann mit Ctrl + D wieder geschlossen werden.

```bash
labstudent@workstation:~$ at now + 1 hour
warning: commands will be executed using /bin/sh
at> echo test
at> <STRG+D>
```

auch mit Pipe möglich:

```bash
echo "tar -czf backup.tar.gz /home/user/docs" | at midnight
```

`<ZEITANGABE>` beschreibt wann ein Job ausgeführt wird. Beispiele:

- `now + 5min`
- `teatime tomorrow`
  - `teatime` ist 16:00
- `noon +4 days`
- `5pm may 12 2022`

Befehl	Beschreibung
`at`	Liest Befehle von der Standardeingabe, die zu einem späteren Zeitpunkt ausgeführt werden sollen.
`at now + 3min`	Startet einen interaktiven Dialog (oder nimmt Pipes entgegen), um Aufgaben in genau 3 Minuten auszuführen.
`atq` / `at -l`	Listet alle aktuell geplanten Aufgaben des Benutzers auf.
`atrm <ID>` / `at -d <ID>`	Löscht einen oder mehrere geplante Jobs anhand ihrer Job-Nummer.

12.16.2. crontab

- Der cron-Daemon kann Skripte und Programme zu einer vorgegebenen Zeit starten
- Befehle werden in der crontab gespeichert
- Es gibt eine systemweite crontab (`/etc/crontab`) und eine für jeden Benutzer (`/var/spool/cron/crontabs`)

Die crontab Tabelle besteht aus sechs Spalten:

- Spalte 1 - 5 dienen der Zeitangabe (Minute, Stunde, Tag, Monat, Wochentage)
- Spalte 6 enthält den Befehl
- Einträge in der systemweitern CronTab Datei haben eine zusätzliche Spalte für den Benutzer

```
┌──────────── Minute (0 - 59)
│ ┌────────── Stunde (0 - 23)
│ │ ┌──────── Tag des Monats (1 - 31)
│ │ │ ┌────── Monat (1 - 12)
│ │ │ │ ┌──── Wochentag (0 - 6) (0 ist Sonntag)
│ │ │ │ │
* * * * * pfad/zum/command
```

Wiederkehrende Jobs sollten wenn möglich über die systemweite crontab Datei erstellt werden

Systemjobst werden auch in `/etc/cron.d/` definiert, dadurch kann die benutzerdefinierte crontab Datei vor der Überschreibung geschützt werden

Befehl	Beschreibung
`crontab -l`	Zeigt alle aktiven geplanten Aufgaben an.
`crontab -e`	Öffnet den Editor, um Aufgaben zu erstellen oder zu ändern.
`crontab -r`	Löscht die komplette Crontab

12.16.2.1. Beispiele

Format	Beschreibung
`0 * * * *`	Jede Stunde (zur Minute 0)
`*/15 * * * *`	Alle 15 Minuten
`0 */2 * * *`	Alle 2 Stunden
`0 18 * * 0-6`	Jeden Tag von Mo-Sa um 18:00 Uhr
`10 2 * * 6,0`	Samstags und sonntags um 02:10 Uhr
`0 0 * * 0`	Jeden Sonntag um Mitternacht

```
#M   S     T M W   Befehl
#-------------------------------------------------------------------
5    9-20 * * *   /home/BENUTZERNAME/script/script1.sh > /dev/null
*/10 *    * * *   /usr/bin/script2.sh > /dev/null 2>&1
59   23   * * 0,4 cp QUELLDATEI ZIELDATEI
*    *    * * *   DISPLAY=:0 LANG=de_DE.UTF-8 zenity --info --text "
    Beispiel für das Starten eines Programms mit GUI"
0    0    * * *   backup
0 1 8-14 1-12 *   [ "$(date +\%u)" == "6" ] && Befehl
#-------------------------------------------------------------------
```

3. Fünf Minuten nach jeder vollen Stunde zwischen 9 und 20 Uhr (also 9:05, 10:05, ..., 20:05)
4. Alle 10 Minuten
5. Jeden Sonntag und Donnerstag um 23:59
6. Jede Minute ein Programm mit GUI, das die Display- und die Sprach-Variable benötigt
7. Jeden Tag Punkt Mitternacht 00:00 Uhr
8. Jeden 2. Samstag im Monat um 01:00 Uhr (Wichtig, das die entsprechende Shell-Variable (hier: /bin/bash) gesetzt ist)

12.16.3. systemd Timer

Timer sind systemd-Unit-Dateien, deren Namen auf `.timer` enden.

- Alternative zu Cron
- Verstehen Kalenderangaben, einfache Zeitereignisse und können asynchron ausgeführt werden.
- Eine Timer-Unit aktiviert eine andere Unit (z. B. einen `.service`), deren Name normalerweise mit dem Timer-Namen übereinstimmt.
- systemd protokolliert Timer-Ereignisse automatisch in den Systemjournalen, was die Fehlersuche erleichtert.

Beispiel einer Konfigurationsdatei:

```ini
[Unit]
Description=Run system activity accounting tool every 10 minutes

[Timer]
OnCalendar=*:00/10

[Install]
WantedBy=sysstat.service
```

12.16.3.1. Zeitangaben

Mit `OnCalendar` Es können sehr spezifische Intervalle definiert werden, z. B. `2019-03-* 12:35,37,39:16`.

- Dies aktiviert den Dienst im März 2019 täglich um 12:35:16, 12:37:16 und 12:39:16 Uhr.

Auch relative Timer möglich (z.B. beim Boot, oder Ähnliches).

Mit `OnUnitActiveSec=15min` wird die Einheit 15 Minuten nach der letzten Aktivierung durch den Timer erneut ausgelöst.

12.16.3.2. Management-Befehle

- Nach **jeder Änderung** an einer Unit-Datei muss der Manager **neu geladen** werden: `systemctl daemon-reload`
- **Timer aktivieren**: `systemctl enable --now <unitname>.timer`
- **Status & Übersicht**:
  - `systemctl status <unitname>.timer` (Status eines Timers prüfen)
  - `systemctl list-timers` (Alle geplanten Timer anzeigen)

12.16.3.3. Wichtige System-Timer

Die Unit `systemd-tmpfiles-clean.timer` wird standardmässig genutzt, um regelmässig temporäre Dateien zu löschen.

12.16.4. Managen der temporären Verzeichnisse

Mit systemd-tmpfiles können temporärer Verzeichnisse und Dateien verwaltet werden

`systemd-tmpfiles-setup` ist eine der ersten systemd Units welche gestartet werden
- Timer Unit verhindert dass Systeme mit langer Ausührungszeit die Festplatte noch mit veralteten Dateien füllt

`systemd-tmpfiles-setup` liest Anweisungen aus:
- `/usr/lib/tmpfiles.d/*.conf`
- `/run/tmpfiles.d/*.conf`
- `/etc/tmpfiles.d/*.conf`

die darin aufgeführten Dateien un Verzeichnisse werden dann erstellt, gelöscht oder mit Berechtigungen gesichert.

12.17. Optimierung der Systemleistung

Die Leistung des Systems kann optimiert werden indem die Geräteeinstellung basierend auf verschiedenen Anwendungfalls-Workloads angepasst werden.

Dafür wird der **Daemon** **`tuned`** wendet Tuning Anpassungen an die entweder statisch oder dynamisch sind:
- statisches Tuning: Festlegung der allgemeinen Leistungserwartung ohne diese anzupassen wenn sich die Aktivitätsebenen ändern
- dynamisches Tuning: tuned überwacht die Systemaktivität und passt die Einstellungen entsprechend an

Das dynamische Tuning ist standardmässig deaktiviert. Variable `dynamic_tuning` in der Konfigurationsdatei `/etc/tuned/tuned-main.conf`

12.17.1. Monitoring

Es gibt drei verschiedenen Monitoring Plug ins:
- disk: Überwacht die Festplattenauslastung basierend auf den I/Os
- net: Überwacht die Netzwerkauslastung anhand der Anzahl übertragenen Pakete
- load: Überwacht die Auslastung der CPUs

12.17.2. Tuning

Es gibt drei verschiedene Tunings Plug ins:
- disk: Legt Festplattenparameter fest (z.B. Festplatten Scheduler, Spin-Down Zeitlimit, erweiterte Energieversorgung)
- net: Konfiguriert Schnittstellen geschwindigkeit und Wake-on-LAN-Funktion
- load: legt CPU Parameter fest (z.B CPU Regler oder Latenz)

12.17.3. tuned

tuned ist standardmässig in RHEL enthalten

tuned Profile werden abgespeichert in:
- `/usr/lib/tuned`
- `/etc/tuned`

Jedes Profil verfügt über eine separater Verzeichnis in dem sich die Konfigurationsdatei `tuned.conf` befindet

Folgende Profile sind enthalten:
- Energiespar Profile
- Performance Profile

Performance Profile enthalten Profile für folgende Aspekte:
- Geringe Latenz zu Speicher und Netzwerk
- Hoher Durchsatz für Speicher und Netwerk
- Leistung des virtualisierten Rechners
- Leistung des Virtualisierungshosts

Beispiel Profile:

Tuned-Profil	Zweck
`balanced`	Ideal für Systeme, die einen Kompromiss zwischen Energieeinsparung und Leistung erfordern.
`powersave`	Optimiert das System für maximale Energieeinsparung.
`throughput-performance`	Optimiert das System für maximalen Durchsatz.
`accelerator-performance`	Führt dieselben Optimierungen wie `throughput-performance` durch und reduziert außerdem die Latenz auf weniger als 100 µs.
`latency-performance`	Ideal für Serversysteme, die eine geringe Latenz erfordern, geht jedoch auf Kosten des Stromverbrauchs.
`network-throughput`	Abgeleitet vom Profil `throughput-performance`. Für den maximalen Netzwerkdurchsatz werden zusätzliche Parameter zur Netzwerkoptimierung angewendet.
`network-latency`	Abgeleitet vom Profil `latency-performance`. Aktiviert zusätzliche Parameter für die Netzwerkoptimierung zur Bereitstellung einer geringen Netzwerklatenz.
`desktop`	Abgeleitet vom Profil `balanced`. Ermöglicht eine schnellere Antwort interaktiver Anwendungen.
`hpc-compute`	Abgeleitet vom Profil `latency-performance`. Ideal für High Performance Computing.
`virtual-guest`	Optimiert das System für maximale Leistung, wenn es auf einem virtuellen Rechner ausgeführt wird.
`virtual-host`	Optimiert das System für maximale Leistung, wenn es als Host für virtuelle Rechner wird.

Befehl	Beschreibung
`tuned-adm active`	Anzeigen des aktiven Profiles
`tuned-adm list`	Anzeigen der verfügbaren Profile
`tuned-adm profile profilename`	Wechseln zu einem anderen Profil

12.18. Prozessplanung beeinflussen

Die Prozessplanung kommt zum Zug wenn es mehr Threads gibt als verfügbare CPU Units.

- Jeder Prozess hat eine andere Priorität
- Scheduler entscheidet welcher Prozess zuerst auf der CPU ausgeführt wird
- Scheduler nutzt Scheduling-Policies
- Scheduling-Policies definieren einfache Regeln zur Organisation von Prozessen und deren Priorisierung
- Scheduling Policies können Prozessprioritäten nutzen, müssen aber nicht
  - Beispiele: Prioritätsscheduling (nutzt Prioritäten), Round Robin (nutzt keine Prioritäten)

Linux verfügt über Scheduling Policies für:
- interaktive Anwendungsanforderungen
- nicht interaktive Batch-Anwendungsverarbeitung (Non Real Time Scheduling)
- Echtzeit Anwendungsanforderungen (Real-Time-Scheduling)

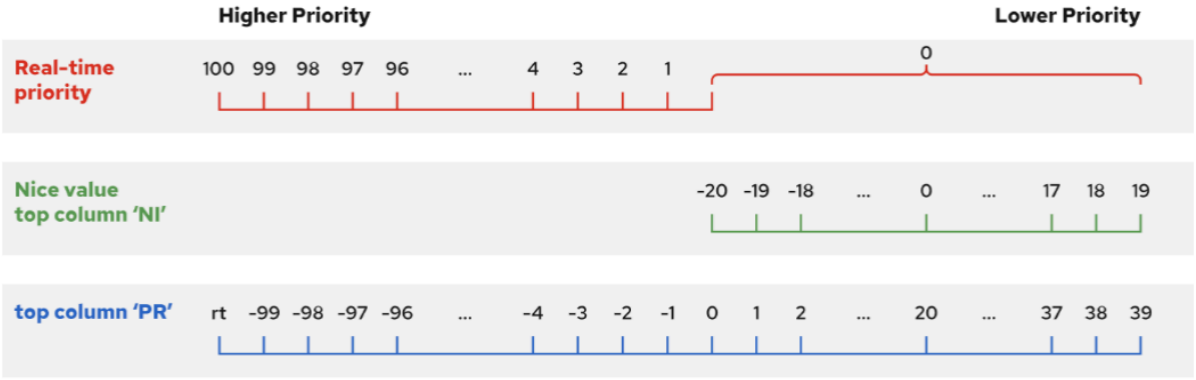
Real Time Scheduling verwendet Prozess Prioritäten

12.18.1. Non Real Time Scheduling

Non Real Time Scheduling verwendet den Completely Fair Schedule (CFS)
- dieser ist in einer binären Suchstruktur organisiert
- funktioniert am besten wenn hoher Datendurchsatz Prioriät hat und eine grosse Anzahl Threads vorhanden ist
- eine dynamische Prioritätslisten wird erstellt welche auf dem Niceness Wert jedes Prozessor Threads basiert
- Administratoren können Niceness wert ändern

12.18.2. Real-Time-Scheduling: Real Time Prozessprioritäten, nice Werten und Prozessprioritäten

- Allen Prozessen wird eine Prioritätswert PR zugeordnet, damit alle Scheduling Policies damit umgehen können
- Je nach dem werden nice-Werte oder Real-Time-Prozessprioritäten zu Prioritätswerten gemappt
- Real Time Prozessprioritäten sind höher angesiedelt als PRs
- Nice Wert von −20 wird auf PR 0 gemappt
- Nice Wert von 19 wird auf PR 39 gemappt



- Real-time priority wird vom Kernel vergeben
- nice level kann manuell gesetzte werden

13. SELinux

SELinux (Security-Enhanced Linux)
- Sicherheitserweiterung des Linux-Kernels
- Umsetzung des FLASK-Sicherheitskonzepts (ursprünglich von der NSA)

Zugriffskontrolle
- Sehr granulare Steuerung von Zugriffen auf Systemressourcen
- Prozesse dürfen nur explizit erlaubte Aktionen ausführen
- Zugriff basiert auf Richtlinien (Policies) und SELinux-Boolean-Flags

Schutzumfang
- Kontrolle von Zugriffen, keine Inhaltskontrolle
- Schutz für Dateien, Prozesse, Netzwerkports und weitere Ressourcen

Richtlinien (Policies)
- Von Anwendungsentwicklern definiert
- Legen fest, welche Aktionen erlaubt sind
- Beschreiben zulässige Zugriffe auf:
  - Programme
  - Konfigurationsdateien
  - Datendateien
  - Netzwerkports

Label-basiertes Modell
- Prozesse und Ressourcen besitzen vordefinierte Labels
- Zugriffsentscheidungen basieren auf Label und Policy, nicht auf Benutzerrechten

Durchsetzung
- Prüfung jeder Zugriffsanfrage im Kernel
- Erlaubte Aktionen werden ausgeführt
- Verweigerte Zugriffe werden im SELinux-Log protokolliert

13.1. Warum Security-Enhanced Linux (SELinux) verwenden?

Präventiver Sicherheitsansatz
- Nicht alle Sicherheitslücken sind im Voraus bekannt
- Erzwingt explizite Zugriffsregeln
- Zusätzliche Sicherheits- und Komplexitätsschicht
- Schutz vor unbekannten Schwachstellen durch **Whitelisting**
  - Nur explizit erlaubte Aktionen sind zulässig
- Sicherheitsprobleme bleiben auf einzelne Systemteile begrenzt
  - Keine unkontrollierte Ausbreitung im System

13.2. SELinux-Modi

Enforcing: SELinux-Sicherheitsrichtlinie wird erzwungen
- Zugriffskontrollregeln werden strikt durchgesetzt
- Standard- und empfohlener Betriebsmodus

Permissive: SELinux gibt Warnung aus, statt sie zu erzwingen
- SELinux ist aktiv, blockiert jedoch keine Zugriffe
- Regelverstösse werden lediglich protokolliert
- Einsatz für Tests und Fehlersuche

Disabled: keine SELinux-Richtlinie geladen
- SELinux vollständig deaktiviert
- Keine Durchsetzung und keine Protokollierung
- **Nicht empfohlen**

Standard RedHat Enterprise Linux: Enforcing
Standard Rocky Linux: Permissive

13.3. Sicherheitsmodell

Klassisches Linux-Sicherheitsmodell
- Zugriffskontrolle über Benutzer, Gruppen und Dateirechte
- Rechtevererbung über den ausführenden Prozess
- Kompromittierte Prozesse erhalten alle Rechte ihres Benutzers

Beispiel Webserver
- Externer Zugriff erfordert offene Firewall-Ports
- Erhöht die Angriffsfläche des Systems
- Kompromittierung des Webserver-Prozesses möglich
- Angreifer erhält Rechte des Webserver-Benutzers (z.B. apache)
  - Lesezugriff auf `/var/www/html`
  - Zugriff auf `/tmp`, `/var/tmp`
  - Zugriff auf weltweit beschreibbare Dateien und Verzeichnisse

Problem ohne SELinux
- Keine zusätzliche Einschränkung kompromittierter Prozesse
- Missbrauch vorhandener Benutzer- und Gruppenrechte möglich

SELinux-Erweiterung des Sicherheitsmodells
- Zusätzliche obligatorische Zugriffskontrolle
- Zugriff nicht nur über Benutzerrechte, sondern über Richtlinien

SELinux-Kontexte
- Alle Prozesse, Dateien, Verzeichnisse und Ports besitzen ein Sicherheitslabel
- Richtlinien definieren:
  - Welcher Prozess
  - auf welche Ressource
  - mit welcher Aktion zugreifen darf
- Standardverhalten: **kein Zugriff ohne explizite Erlaubnis**

13.3.1. Dateikontext

- Jede **Datei** besitzt einen SELinux-Sicherheitskontext
- Kontext besteht aus mehreren Bestandteilen:
  - SELinux-User
  - Rolle
  - Typ
  - Level
- Beispiel:
  - `unconfined_u:object_r:httpd_sys_content_t:s0/var/www/html/file2`

Typ-basierte Zugriffskontrolle

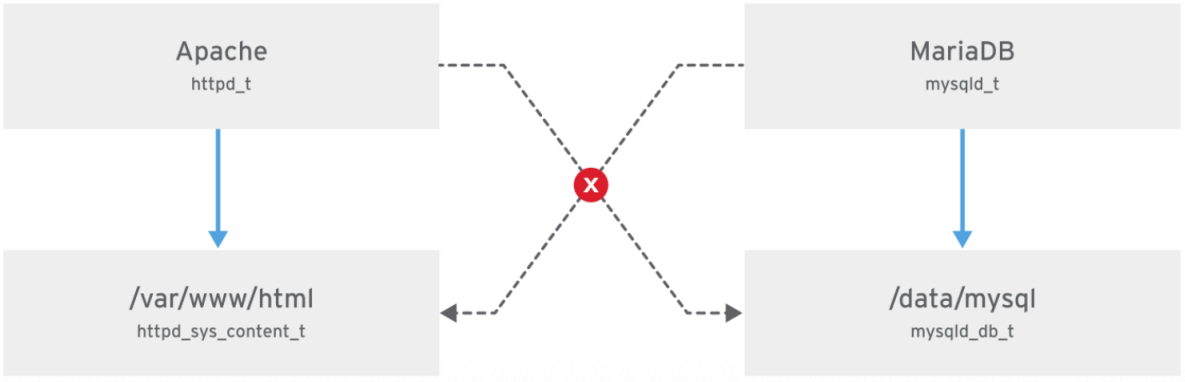
- Der **Typ** ist das zentrale Element für Zugriffsentscheidungen
- Prozesse dürfen nur auf Dateien mit erlaubtem Typ zugreifen

Webserver-Beispiel

- Webserver-Prozess läuft im Kontext `httpd_t`
- Webinhalte besitzen standardmässig den Typ:
  - `httpd_sys_content_t`
- Dateien unter `/var/www/html` sind entsprechend gelabelt

Trennung von Diensten

- Apache (`httpd_t`) darf auf Webinhalte zugreifen
- MariaDB (`mysqld_t`) darf auf Datenbankverzeichnisse zugreifen
  - z.B. `/data/mysql` mit Typ `mysqld_db_t`
- Direkter Zugriff zwischen Diensten ist verboten



Default-Deny-Prinzip

- Ohne explizite SELinux-Regel ist kein Zugriff erlaubt
- Erhöhte Sicherheit bei kompromittierten Prozessen

Anzeige von SELinux-Kontexten

- Viele Linux-Befehle unterstützen die Option `-Z`
- Anzeige oder Setzen von SELinux-Kontexten möglich
- Typische Befehle mit `-Z` :
  - `ps -Z` ⟶ Prozesskontexte
  - `ls -Z` ⟶ Dateikontexte
  - `cp -Z` , `mkdir -Z` ⟶ Kontext beim Erstellen setzen

Prozesskontexte
- Jeder laufende Prozess besitzt einen SELinux-Kontext
- Beispiel:
  - Apache läuft im Typ `httpd_t`
- Anzeige z.B. mit:
  - `ps -ZC httpd`

Dateikontexte
- Dateien und Verzeichnisse besitzen definierte Typen
- Beispiel Webserver:
  - `/var/www/html` → `httpd_sys_content_t`
- Trennung von ausführbaren Skripten und Inhalten:
  - `httpd_sys_script_exec_t`
  - `httpd_sys_content_t`

Ändern des SELinux-Modus zur Laufzeit
- Aktuellen Modus anzeigen:
  - `getenforce`
- Modus ändern:
  - `setenforce Enforcing`
  - `setenforce Permissive`
  - `setenforce 1` / `setenforce 0`

SELinux-Modus beim Booten festlegen
- Übergabe von Kernel-Parametern:
  - `enforcing=1` → Enforcing
  - `enforcing=0` → Permissive

13.4. Festlegen des SELinux-Standardmodus

SELinux kann dauerhaft über die Datei `/etc/selinux/config` konfiguriert werden.

In der Standardkonfiguration ist der Modus auf `enforcing` gesetzt.

Alternativ sind die Modi `permissive` und `disabled` möglich, welche in den Kommentaren der Datei dokumentiert sind.

```ini
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing  - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled   - No SELinux policy is loaded.
SELINUX=enforcing

# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     minimum  - Modification of targeted policy. Only selected
#                processes are protected.
#     mls      - Multi Level Security protection.
SELINUXTYPE=targeted
```

13.5. Wichtigste Befehle

Befehl	Wirkweise
`getenforce`	Zeigt den aktuell aktiven SELinux-Modus (**Enforcing**, **Permissive** oder **Disabled**).
`setenforce 1 | 0` `setenforce Enforcing | Permissive`	Schaltet SELinux zur Laufzeit auf **Enforcing** (1) oder **Permissive** (0); Änderung ist nicht persistent.

Befehl	Wirkweise
`ps -Z`	Zeigt den SELinux-Kontext laufender Prozesse an.
`ls -Z`	Zeigt den SELinux-Dateikontext von Dateien und Verzeichnissen an.
`semanage fcontext -l`	Listet alle definierten Regeln für Standard-Dateikontexte auf.
`semanage fcontext -a -t <typ> <pfad>`	Fügt eine Regel hinzu, welche den SELinux-Typ für Dateien anhand eines Pfads festlegt.
`restorecon <pfad>`	Setzt den SELinux-Kontext einer Datei oder eines Verzeichnisses auf den Standardwert zurück.
`restorecon -RV <pfad>`	Setzt Dateikontexte rekursiv und ausführlich für ein Verzeichnis gemäss den fcontext-Regeln zurück.
`chcon -t <typ> <datei>`	Ändert den SELinux-Typ einer Datei temporär; Änderung geht bei `restorecon` verloren.
`getsebool -a`	Listet alle verfügbaren SELinux-Booleans und deren aktuellen Status auf.
`setsebool <boolean> on | off`	Aktiviert oder deaktiviert ein SELinux-Boolean temporär.
`setsebool -P <boolean> on | off`	Setzt ein SELinux-Boolean dauerhaft (persistent über Neustarts).
`sealert -l "*"`	Zeigt aufbereitete Hinweise und Erklärungen zu SELinux-Fehlern an.
`ausearch -m AVC -ts recent`	Durchsucht das Audit-Log nach aktuellen SELinux-Zugriffsverweigerungen (AVC-Meldungen).

13.6. Boolesche SELinux-Werte

- SELinux-Richtlinien werden von Anwendungsentwicklern definiert
- Richtlinien können optionales Verhalten enthalten
- Dieses Verhalten kann systemabhängig aktiviert oder deaktiviert werden

SELinux-Booleans

- Schalten optionales Verhalten innerhalb einer SELinux-Richtlinie
- Aktivieren oder deaktivieren einzelne Regeln
- Umsetzung eines konfigurierbaren Sicherheitsverhaltens

Eigenschaften

- Anwendungsspezifisch (z.B. Webserver, Datenbank)
- Müssen gezielt pro Anwendung ausgewählt werden
- Erlauben feine Anpassungen ohne Änderung der Policy

Verwaltung

- Anzeige aller Booleans:
  - `getsebool -a`
- Setzen eines Booleans:
  - `setsebool <boolean> on | off`
- Dauerhaftes Setzen:
  - `setsebool -P <boolean> on | off`

Dokumentation

- Service-spezifische Booleans sind in den SELinux-Manpages des jeweiligen Dienstes beschrieben

13.7. Beheben von SELinux-Problemen

Grundidee

- Fehlfunktionen von Anwendungen entstehen häufig durch SELinux-Zugriffsverweigerungen

- Zur Analyse stehen spezielle Tools und Logs zur Verfügung
- Grundverständnis der SELinux-Arbeitsweise ist Voraussetzung für die Fehlerbehebung

Aufbau von SELinux-Richtlinien
- SELinux besteht aus gezielten Richtlinien mit expliziten Erlaubnissen
- Ein Richtlinieneintrag definiert:
  - einen gekennzeichneten Prozess (Prozesstyp)
  - eine gekennzeichnete Ressource (Datei-, Verzeichnis- oder Portkontext)
  - eine explizit erlaubte Aktion
- Aktionen können sein:
  - Systemaufrufe
  - Kernel-Funktionen
  - andere spezifische Programmieroperationen

Default-Deny-Prinzip
- Existiert keine Regel für die Kombination aus Prozess, Ressource und Aktion:
  - Zugriff wird verweigert
- Verweigerte Aktionen werden mit detaillierten Kontextinformationen protokolliert

13.7.1. Anmerkungen zur Fehleranalyse

Zugriffsverweigerungen
- Die meisten SELinux-Blockierungen zeigen korrektes Verhalten an
- Ziel ist Sicherheit, nicht Fehlkonfiguration per se

Häufige Ursachen
- Falscher SELinux-Kontext bei:
  - neu erstellten
  - kopierten
  - verschobenen Dateien
- Besonders häufig bei Webserver- und Datenbankdaten

Behebung
- Dateikontexte können meist einfach korrigiert werden
- Voraussetzung:
  - Es existiert bereits eine passende Richtlinie für den Speicherort
- Typisches Mittel:
  - Wiederherstellen des Standardkontexts (z.B. mit `restorecon`)

Boolesche Richtlinien
- Optionale SELinux-Funktionen sind über Booleans steuerbar
- Dokumentation in den jeweiligen `_selinux`-Manpages der Services
- Aktivierung erfordert oft zusätzliche Nicht-SELinux-Konfiguration

Wichtiger Hinweis
- SELinux ersetzt keine klassischen Dateiberechtigungen oder ACLs
- Beide Sicherheitsmechanismen wirken ergänzend

13.8. SELinux AVC-Analyse
- SELinux prüft bei jeder Aktion Zugriffe anhand von Access Vectors (AV)
- Entscheidung erfolgt über den Access Vector Cache (AVC)
- Bei **verweigertem Zugriff wird eine AVC-Meldung** erzeugt

13.8.1. Analyse von AVC-Meldungen
- Ursprungsdaten liegen in `/var/log/audit/audit.log`
- `ausearch`
  - Sucht rohe Audit- und AVC-Meldungen
  - Nützlich zur Analyse der exakten Ursache
  - Beispiel:

  - `ausearch -m AVC -ts recent`

- `sealert`
  - Wertet AVC-Meldungen aufbereitet aus
  - Zeigt Ursache, Kontext und mögliche Fixes
  - Beispiel:
    - `sealert -a /var/log/audit/audit.log`

13.9. Portbezeichnung in SELinux

Netzwerkports werden in SELinux ebenfalls mit einem eigenen SELinux-Kontext gekennzeichnet.

SELinux steuert den Netzwerkzugriff, indem Ports mit Kontexten versehen und diese in den Policy-Regeln der Services berücksichtigt werden.

Beispiele:
- Port 22/TCP → `ssh_port_t`
- Port 80/TCP bzw. 443/TCP → `http_port_t`

Schutzmechanismen:
- Ein Prozess darf einen Port nur öffnen, wenn dies durch die Policy erlaubt ist
- Die Bindung erfolgt zwischen Prozesstyp und Portkontexttyp
- Verhindert, dass illegitime Services fremde oder kritische Ports öffnen
- Schutz vor dem Missbrauch legitimer Netzwerkdienste

13.9.1. Verwalten von Portbezeichnungen

- SELinux kontrolliert, welche Services welche Ports überwachen (binden) dürfen
- Nicht standardmässige Ports werden blockiert, wenn sie nicht korrekt gekennzeichnet sind
- Lösung: Port muss mit dem passenden Portkontext in der SELinux-Policy versehen werden
- In der targeted-Policy sind gängige Ports bereits korrekt vorkonfiguriert
- Beispiel: Port 8008/TCP ist meist mit http_port_t für Webserver gekennzeichnet
- Ein Port darf genau einen SELinux-Portkontext haben
- Falscher oder fehlender Kontext → Service darf den Port nicht nutzen

13.9.2. Auflisten von Portbezeichnungen

Datei `/etc/services` enthält Zuordnungen von Servicenamen zu Portnummern

Filtern nach Service:
- `grep <service> /etc/services`

Aktuelle SELinux-Portzuweisungen anzeigen:
- `semanage port -l`

Filtern der Ausgabe:
- Nach Servicename: `semanage port -l | grep <name>`
- Nach Portnummer: `semanage port -l | grep -w <port>`

Eine Portbezeichnung kann für mehrere Protokolle (TCP/UDP) existieren.

13.9.3. Verwalten von Portbindungen

Verwaltung von Portzuweisungen erfolgt mit `semanage port`

Befehl	Wirkweise
`semanage port -a -t <port_type> -p tcp|udp <port>`	Neuen Port einem bestehenden SELinux-Porttyp zuweisen
`semanage port -l -C`	Lokale Änderungen gegenüber der Standard-Policy anzeigen
`semanage port -d -t <port_type> -p tcp|udp <port>`	Portbindung löschen

Befehl	Wirkweise
`semanage port -m -t <new_port_type> -p tcp|udp <port>`	Portbindung ändern (effizienter als Löschen + Hinzufügen)

13.9.4. Firewall

```sh
[root@server-010 ~]# firewall-cmd --zone public --add-port=80/tcp --permanent
```

- Erlaubt Netzwerkverbindungen auf Port `80 (TCP)` in der Zone `public`
- Das Flag `--permanent` sorgt dafür, dass die Regel dauerhaft gespeichert wird

```sh
[root@server-010 ~]# firewall-cmd --reload
```

- Lädt die Firewall-Konfiguration neu, um die Änderungen sofort zu aktivieren

13.10. SELinux-Manpages

Servicespezifische SELinux-Manpages werden nach dem Muster `<service>_selinux` benannt

Diese Manpages enthalten Informationen zu:
- SELinux-Typen
- Boolean-Optionen
- Porttypen

Die SELinux-Manpages sind standardmässig nicht installiert

Installation der Dokumentation:

```bash
dnf install selinux-policy-doc
man -k _selinux
```

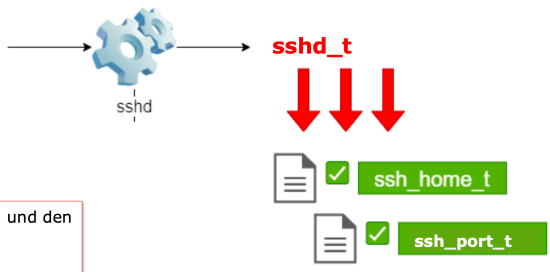
13.11. SELinux: Dateizugriffsverwaltung versus Portverwaltung

Beispiel/Szenario:

- Prozess: *ssh*
- Genutzte Dateien: .ssh/id_rsa
- Genutzte Port: *TCP/22*

SELinux-Kontexte

- Prozesskontext: sshd → *sshd_t*
- Dateikontext: .ssh/id_rsa → *sshd_home_t*
- Portkontext: TCP/22 → *ssh_port_t*



Für den Zugriff von *sshd* auf die Datei und den Port muss eine Policy existieren, die

a) sshd_t → sshd_home_t

b) sshd_t → ssh_port_t

erlaubt.

14. Verwaltung von Festplatten und Blockgerätemanagement

14.1. Blockgeräte und Mouten

14.1.1. Identifizieren von Dateisystemen und Geräten

In Linux gibt es nur einen einzigen Verzeichnisbaum, der bei `/` beginnt. Alle Festplatten, USB-Sticks, SSDs usw. müssen in diesen **Dateibaum** eingefügt werden = Mounten. Typisches Verzeichnis: `/mnt`

Vorteile: Blockgeräte (wie externe Festplatten) lassen sich mit Bordmitteln (`ls` , `cp`) verwalten.

Mountpoint = `<Verzeichnis>` , leeres Verzeichnis (dort wird das Dateisystems eines Geräts sichtbar gemacht), z.B. `/mnt` **Gerät**: `/dev/XY`

14.1.1.1. Benennung von Blockgeräten

Typ des Geräts	Gerätebenenungsmuster
SATA / SAS / USB	`/dev/sda, /dev/sdb ...`
virtuelle Maschine (virtio-blk)	`/dev/vda, /dev/vdb ...`
NVMe-SSDs	`/dev/nvme0, /dev/nvme1 ...`
SD/MMC/eMMC (SD-Karten)	`/dev/mmcblk0, /dev/mmcblk1`

14.1.2. Laufwerkspartitionierungen

- Partitionen sind eigenständige Blockgeräte
- SATA angeschlossene Partitionen, Bennungssystem:
  - erste Partition auf dem erstes Laufwerk `/dev/sda1`
  - dritte Partition auf dem zweiten Laufwerk `/dev/sdb3`
  - usw.
- Paravirtualisierte Speichergeräte haben ein ähnliches Benennungssystem

14.1.3. Befehle

- `df` : Übersicht der lokalen und Remote-Dateisystemgeräte
  - zeigt gesamten Speicherplatz, belegten Speicherplatz, freien Speicherplatz
- Optionen
  - `-h` : Berichte in Kib (210), MiB (220) oder GiB (230)
  - `-H` : Berichte in SI-Einheiten: KB (103), MB (106), GB (109) usw.

```bash
[user@host ~]$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs          912584       0    912584   0% /dev
tmpfs             936516       0    936516   0% /dev/shm
tmpfs             936516   16812    919704   2% /run
tmpfs             936516       0    936516   0% /sys/fs/cgroup
/dev/vda3        8377344 1411332   6966012  17% /
/dev/vda1        1038336  169896    868440  17% /boot
tmpfs             187300       0    187300   0% /run/user/1000
```

- `du` : Grösse eines Verzeichnisses anzeigen
- Optionen
  - `-h`
  - `-H`

```bash
[root@host ~]# du /usr/share
...output omitted...
176 /usr/share/smartmontools
184 /usr/share/nano
8 /usr/share/cmake/bash-completion
8 /usr/share/cmake
356676 /usr/share
```

```bash
[root@host ~]# du -h /usr/share
...output omitted...
176K    /usr/share/smartmontools
184K    /usr/share/nano
8.0K    /usr/share/cmake/bash-completion
8.0K    /usr/share/cmake
369M    /usr/share
```

- Beispiele
  - Prüfen von Dateisystemen: `df -h /dev/xy`
  - Datenträgerbelegung für das Verzeichnis: `du -h <Verzeichnis>`

14.1.3.1. Demo

Befehl	Erklärung
`/dev`	Verzeichnis, das **Gerätedateien** enthält (z. B. Festplatten, USB-Sticks als Blockgeräte).
`lsblk`	Zeigt alle **Blockgeräte** (Festplatten, Partitionen, Mountpoints) übersichtlich an.
`mkfs.ext3 /dev/vdb`	Erstellt ein **ext3-Dateisystem** auf dem Blockgerät `/dev/vdb` (löscht vorhandene Daten).
`mount /dev/vdb /mnt/disk2`	Bindet das Dateisystem von `/dev/vdb` in das Verzeichnis `/mnt/disk2` ein.
`lsblk -fp`	Zeigt Blockgeräte mit **vollständigem Pfad**, **Dateisystem**, **UUID** und **Mountpoint** an.
`mount UUID="1234-1231-234243" /mnt/disk2`	Mountet ein Dateisystem **über seine UUID** (robuster als Gerätenamen).
`umount /mnt/disk2`	Hängt das Dateisystem von `/mnt/disk2` wieder aus.
`lsof /mnt/disk2`	Zeigt alle **Prozesse**, die noch Dateien auf `/mnt/disk2` geöffnet haben (hilfreich bei „device busy“).

14.1.4. Übung: Partitionieren von Festplattenspeicher und Einhängen der Partition (mounten)

→ mit root (`sudo -i`)

Partionierung
auf der Festplatte `/dev/vdb` , Typ `msdos`

```bash
parted /dev/vdb mklabel msdos
```

neue Partition mit Grösse von 1 GB hinzufügen, Sektor 2048, Typ XFS → interaktiver Modus oder mit Befehl möglich

```bash
parted /dev/vdb #interaktiv
parted /dev/vdb mkpart primary xfs 2048s 1001MB #Befehl
```

Überprüfung mit:

```bash
parted /dev/vdb print
```

Partition registrieren:

```bash
udevadm settle
```

Formatierung und Einhängen (mounten)

```bash
mkfs.xfs /dev/vdb1
```

Dauerhaft mounten unter `/archive`:

```bash
mkdir /archive
```

UUID von `/dev/vdb1` ermitteln und kopieren:

```bash
lsblk --fs /dev/vdb
```

in der Datei `/etc/fstab` einen Eintrag hinzufügen

```txt
UUID=e3db1abe-6d96-4faa-a213-b6f85dcc1 /archive xfs defaults 0 0
```

systemd aktualisieren:

```bash
systemctl daemon-reload
```

mounten:

```bash
mount /archive/
```

Überprüfung:

```bash
mount | grep /archive
```

Automatisch Einhängen

Server neu booten (nach ein paar Minuten wieder anmelden)

```bash
systemctl reboot
```

nochmals überprüfen:

```bash
mount | grep /archive
```

14.2. Logische Volumes erstellen und erweitern

14.2.1. Logical Volume Manager (LVM)

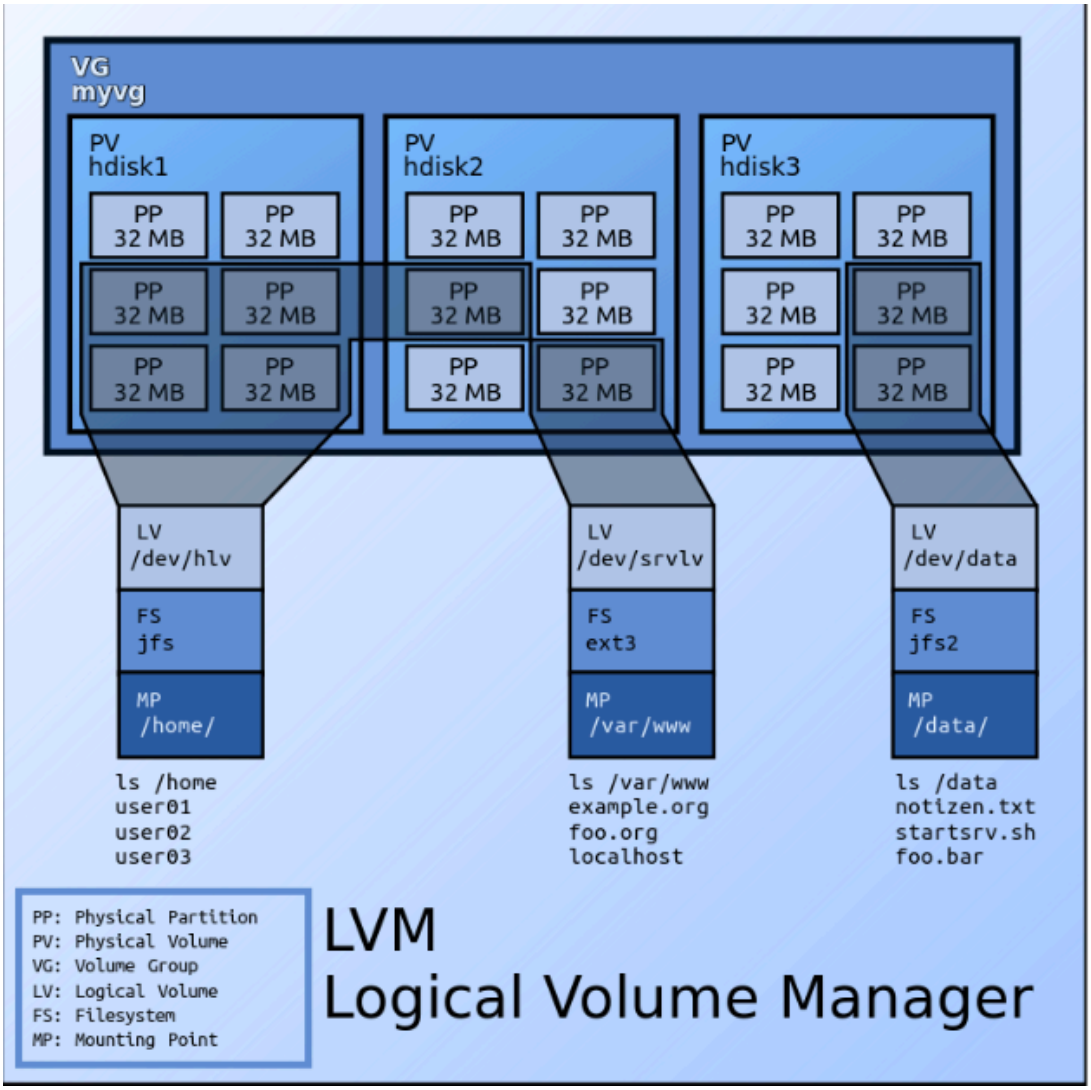
Problem ohne LVM:

- Festplatte -> Partition -> Dateisystem -> Moutmoint
- Partitionen haben feste Grösse
- kann zu Downtime führen
- eher unflexibel

LVM: legt eine flexible Schicht zwischen Hardware und Dateisystem, Abstraktionsebene zwischen Festplatten, Partitionen und Dateisystemen

- grössere Flexibilität
- verbirgt Hardwarekonfiguration des Speichers vor der Software
- man kann die Grösse von Volumens ändern, ohne Anwendungen zu stoppen oder Dateisysteme zu unmounten
- LVM bietet umfassende Befehlszeilentools für die Speicherverwaltung

⟶ Viele LVMs unterstützen die Organisation der Logical Volumes als RAID-Verbund, sodass die Daten gegen Plattenausfälle geschützt werden können oder der Zugriff beschleunigt wird.



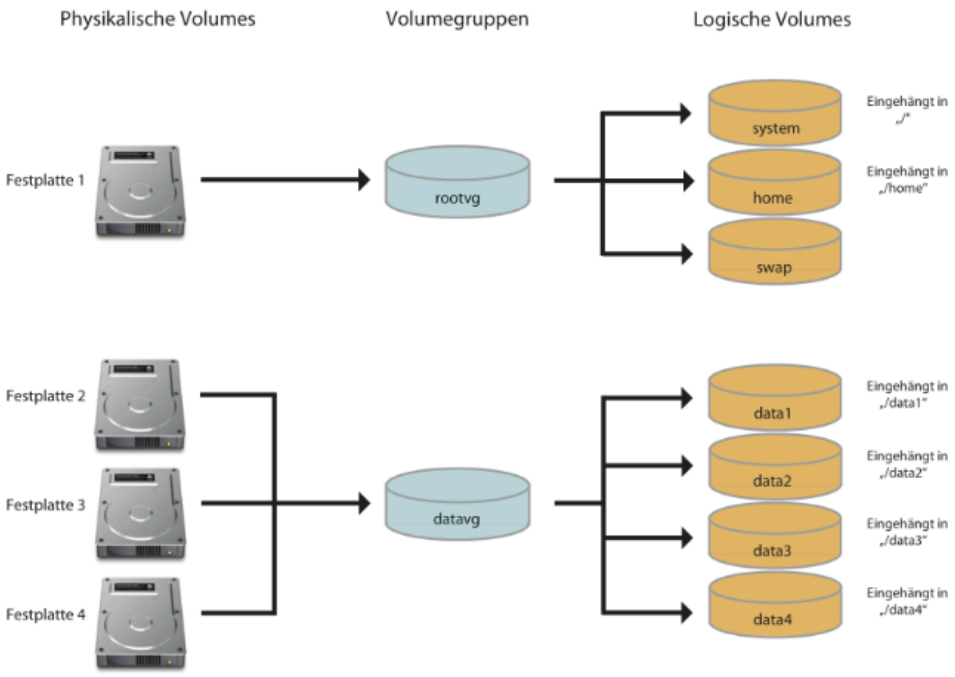
14.2.2. Funktionalität und Vorteile von LVM

Vorteile in Bezug auf Skalierbarkeit und Speicherplatzverwaltung:

- Dynamische Grössenänderung von Volumes ⟶ Keine „Festplatten“ fixer Grösse

- Verteilen von einem Volume über mehrere physische Festplatten im PC ⟶ Zusammenfassen von physischen Festplatten
- Live-Kapazitätserweiterung ⟶ Dynamische Verteilung von Speicherplatz
- Unterstützung für Snapshots ⟶ Sicherungskopie des Dateisystems zu einem bestimmten Zeitpunkt

Die Grösse dieser virtuellen Datenträger lässt sich auch nach dem Anlegen eines Dateisystems noch ändern, selbst wenn schon Daten darin gespeichert wurden oder der Speicher in Benutzung ist.



14.2.2.1. Snapshots in der Praxis

Copy-on-Write (COW): Ein Snapshot speichert nur die Datenblöcke, die sich seit der Erstellung im Original-Volume geändert haben.

Speicherbedarf: Ein Snapshot benötigt freie Extents in der Volume-Gruppe. Wenn der Platz für die Änderungen nicht ausreicht, wird der Snapshot unbrauchbar. +1

Wiederherstellung: Daten werden durch einfaches Kopieren (`cp`) vom gemounteten Snapshot zurück in das Original-Volume wiederhergestellt

14.2.3. Betriebssysteme mit LVM-Unterstützung

Seit 1999 im Linux-Kernel enthalten, LVM basiert auf Device Mapper (DM) und wird nun von jeder modernen Distribution unterstützt.

- **AIX, HP-UX, Tru64 UNIX**: Komplette LVM-Unterstützung mit Spiegelung und Online-Vergrösserung von Logical Volumes.
- **Linux**: LVM-Implementation (1998 von Heinz Mauelshagen), die sich in der Bedienung stark an HP-UX anlehnt. Eine Online-Vergrösserung ist unter anderem mit folgenden Dateisystemen möglich: ext2, ext3, und ext4, JFS, ReiserFS v3 sowie mit XFS.
- **Windows 2000 und höher**: Hier entspricht das logische Volume Management in etwa der Verwaltung von Dynamischen Datenträgern.
- **macOS**: Seit Mac OS X Lion ist ein LVM unter dem Namen CoreStorage implementiert. Apples sogenanntes Fusion Drive verbindet dadurch eine SSD und eine Festplatte zu einem logischen Laufwerk, bei dem die SSD als Pufferspeicher verwendet wird.

14.2.4. Local Volume Manager - Überblick

Physische Geräte:

- physische Geräte zum Speichern von Daten (Bsp. Festplatten, Patitionen, RAID-Arrays, SAN-Datenträger)
- ein physisches LVM-Volume verwendet das gesamte physische Geräte

Physische Volumens (PVs):

- LVM verwendet das ganze physische Geräte als physisches LVN-Volume

Volume-Gruppe (VGs):

- Speicherpools, die aus mind. einem PV bestehen
- funktionale Äquivalent eines gesamten Datenträgers im Vergleich zum physischen Speicher -> nur virtuell, bestehend aus PVs
- **Struktur:** Eine VG besteht aus mindestens einem PV. Ein PV gehört immer genau zu einer VG.
- **Unterteilung:** Die VG wird intern in kleine Einheiten unterteilt (**Physical Extents (PEs)**, Grösse wird automatisch festgelegt, kann aber auch manuell erfolgen
- **Funktion:** Die VG bietet Platz für ungenutzten Speicher oder für mehrere Logische Volumes

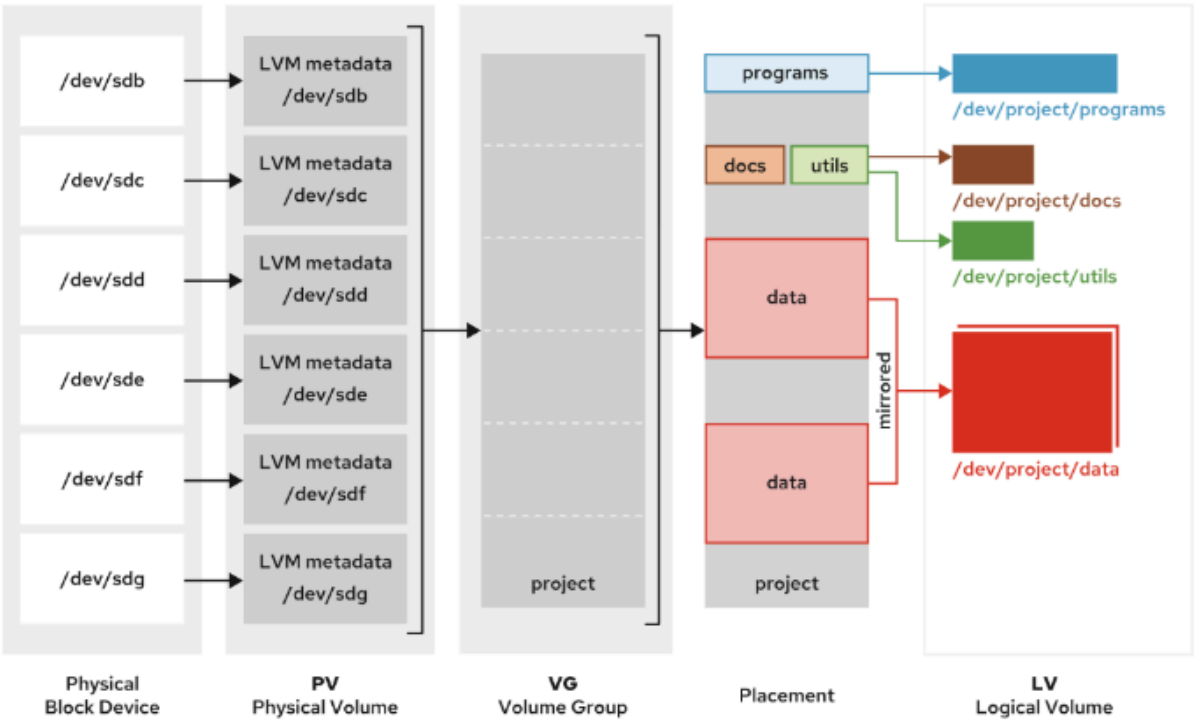
Logisches Volumens (LVs):

- Bereich, welches der User oder das OS nutzt
- **Zusammensetzung:** Ein LV besteht aus einer Sammlung von PEs, die aus der Volume-Gruppe stammen.
- **Besonderheit:** Durch Optionen wie „Spiegelung“ kann festgelegt werden, dass ein LV gleichzeitig mehreren PEs zugeordnet wird (zur Datensicherheit).

14.2.5. Logical Volume Manager-Workflow

Erstellung von LVM-Speicher:

1. **physische Gerätepartitionen** erstellen (optional), `parted`
2. **physische Volumens** (PVs) aus physischen Gerätepartitionen erstellen, `pvcreate`
3. **Volume-Gruppe** (VG) aus mehreren physischen Volumes erstellen, `vgcreate`
4. **logische-Volumes** (LV) aus dem verfügbaren Speicherplatz in der Volume-Gruppe erstellen, `lvcreate`
5. logische-Volume formatieren und mounten oder als Swap-Speicher aktivieren oder unformatierte Volume an eine DB oder Speicherserver für erweiterte Strukturen übergeben
   - Dateisystem auf LV erstellen: z.B: mit `mkfs.ext4`
   - Mountpoint erstellen, z.B. mit `mkdir /archiv`
   - fstab editieren (automatisches mounten) in `/etc/fstab`



14.2.5.1. 1. Erstellen von physischen Gerätepartitionen

`parted /dev/vdb mklabel gpt mkpart primary ...`: Partition erstellen `parted /dev/vdb set 1 lvm on`: Partition als LVM markieren

```bash
[root@host ~]# parted /dev/vdb mklabel gpt mkpart primary 1MiB 769MiB
...output omitted...
[root@host ~]# parted /dev/vdb mkpart primary 770MiB 1026MiB
[root@host ~]# parted /dev/vdb set 1 lvm on
[root@host ~]# parted /dev/vdb set 2 lvm on
[root@host ~]# udevadm settle
```

`udevadm settle`: Dieser Befehl ist im Workflow entscheidend.

- Er lässt das System warten, bis der Kernel die neuen Partitionen registriert hat, bevor mit `pvcreate` fortgefahren wird.

14.2.5.2. 2. Erstellen von physischen Volumes

`pvcreate /dev/vdb1 /dev/vdb2`

```bash
[root@host ~]# pvcreate /dev/vdb1 /dev/vdb2
Physical volume "/dev/vdb1" successfully created.
Physical volume "/dev/vdb2" successfully created.
Creating devices file /etc/lvm/devices/system.devices
```

Ohne Name oder Label. Gesamtes Device wird inkludiert und initialisiert.

14.2.5.3. 3. Erstellen von Volume-Gruppen

`vgcreate vg01 /dev/vdb1 /dev/vdb2`

```bash
[root@host ~]# vgcreate vg01 /dev/vdb1 /dev/vdb2
Volume group "vg01" successfully created
```

Neues virtuelles Gerät `vg01`.

14.2.5.4. 4. Erstellen von logischen Volumes

`lvcreate -n lv01 -L 300M vg01`: Erstellt ein LV mit einer Grösse von 128 MiB, gerundet auf das nächste PE.

`lvcreate -n lv01 -l 32 vg01`: Erstellt ein LV mit einer Grösse 128 MiB aus 32 PEs mit jeweils 4 MiB.

Der Befehl schlägt fehl, wenn die VG nicht genügend freie Physical Extents (PEs) hat. Die LV-Grösse wird auf den nächsten PE-Wert aufgerundet, wenn sie nicht genau übereinstimmt.

```bash
[root@host ~]# lvcreate -n lv01 -L 300M vg01
Logical volume "lv01" created.
```

14.2.6. Dateisystem auf dem logischen Volume erstellen

logische Volume mit üblichen Namen `/dev/vgname/lvname` oder mit dem Namen des Kernel-Device Mapper `/dev/mapper/_vgname-lvname_` verwenden

- `mkfs` : auf dem neuen logischen Volume ein Dateisystem erstellen
- `mkdir` : Mount-Punkt erstellen
- Eintrag der Datei `/etc/fstab` hinzufügen, damit das Dateisystem dauerhaft verfügbar ist
- `mount` : LV bereitstellen

Hinweis: logisches Volume kann anhand des Namens oder der UUID gemountet werden (da LVM die PVs parst, die nach der UUID suchen). Ist auch dann möglich, wenn das VG mit einem Namen erstellt wurde, da das PV immer eine UUID enthält.

```bash
[root@host ~]# mkfs -t xfs /dev/vg01/lv01
...output omitted...

[root@host ~]# mkdir /mnt/data

# Eintrag für die /etc/fstab (Dauerhaftes Mounten):
/dev/vg01/lv01 /mnt/data xfs defaults 0 0

[root@host ~]# mount /mnt/data/
```

14.2.7. Unterstützung für Virtual Data Optimizer (VDO) in LVM

VDO ist eine Software die folgendes für Speicher bereitstellt:

Deduplizierung: Technik zur Reduzierung des Verbrauchs von Speicherressourcen. Es werden mehrere Kopien von doppelten Blöcken eliminiert.

Komprimierung: nimmt einzelne eindeutige Blöcke und schrumpft diese. Diese reduzierten Blöcke werden dann effizient in physische Blöcke zusammengepackt.

Thin Provisioning: verwaltet Zuordnung von logischen Blöcken, die von VDO präsentiert werden, zu dem Ort, an dem die Daten tatsächlich physisch gespeichert wurden. Eliminiert auch alle Blöcke, die nur aus Nullen bestehen.

14.2.7.1. Verwendung

- Paket `vdo` und `kmod-kvdo` installieren
  - `dnf install vdo kmod-kvdo`
- VDO-LV erstellen: `lvcreate --type vdo --name xy --size ....`

```sh
[root@host ~]# lvcreate --type vdo --name vdo-lv01 --size 5G vg01
  Logical blocks defaulted to 523108 blocks.
  The VDO volume can address 2 GB in 1 data slab.
  It can grow to address at most 16 TB of physical storage in 8192 slabs.
  If a larger maximum size might be needed, use bigger slabs.
  Logical volume "vdo-lv01" created.
```

14.2.7.2. Status Anzeige

LVM umfasst verschiedene Dienstprogramme zum Anzeigen der Statusinformationen von PV, VG und LV.

`pvdisplay`

```
[root@host ~]# pvdisplay /dev/vdb1
  --- Physical volume ---
  PV Name               /dev/vdb1                     (1)
  VG Name               vg01                          (2)
  PV Size               731.98 MiB / not usable 3.98 MiB (3)
  Allocatable           yes
  PE Size               4.00 MiB                      (4)
  Total PE              182
  Free PE               107                           (5)
  Allocated PE          75
  PV UUID               zP0gD9-NxTV-Qtoi-yfQD-TGpL-0Yj0-wExh2N
```

1. Mit `PV Name` wird der Gerätename angezeigt.
2. Mit `VG Name` wird die Volume-Gruppe angezeigt, der das PV zugewiesen ist.
3. Mit `PV Size` wird die physische Größe des PV angezeigt, einschließlich des nicht verwendbaren Speicherplatzes.
4. Mit `PE Size` wird die Größe des physischen Extents angezeigt.
5. Mit `Free PE` wird die PE-Größe angezeigt, die in der VG verfügbar ist, um neue LVs zu erstellen oder vorhandene LVs zu erweitern.

`vgdisplay`

```
[root@host ~]# vgdisplay vg01
  --- Volume group ---
  VG Name               vg01                    (1)
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               1012.00 MiB             (2)
  PE Size               4.00 MiB
  Total PE              253                     (3)
  Alloc PE / Size       75 / 300.00 MiB
  Free  PE / Size       178 / 712.00 MiB        (4)
  VG UUID               jK5M1M-Yvlk-kxU2-bxmS-dNjQ-Bs3L-DRlJNc
```

1. Mit `VG Name` wird der Name der Volume-Gruppe angezeigt.
2. Mit `VG Size` wird die Gesamtgröße des Storage-Pools angezeigt, der für die LV-Zuweisung zur Verfügung steht.
3. Mit `Total PE` wird die Gesamtgröße der PE-Units angezeigt.
4. Mit `Free PE / Size` wird der Speicherplatz angezeigt, der in der VG verfügbar ist, um neue LVs zu erstellen oder vorhandene LVs zu erweitern.

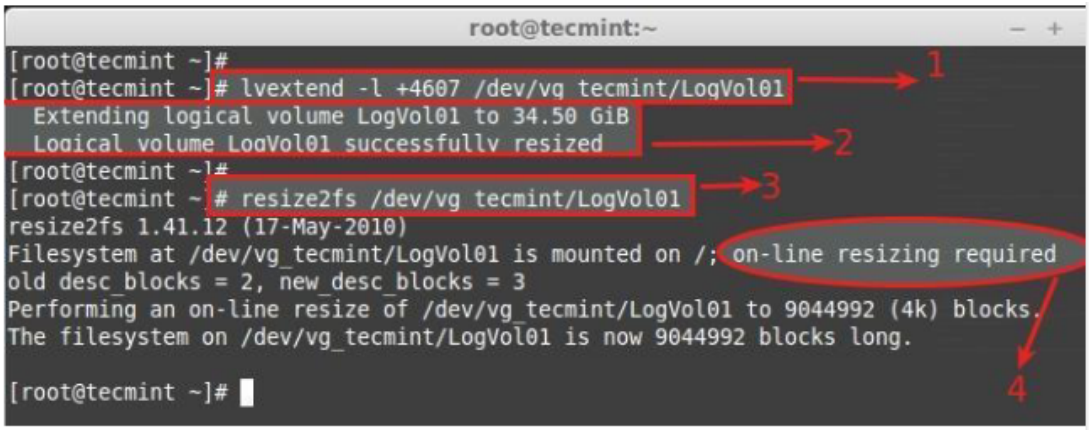
`lvdisplay`

```
[root@host ~]# lvdisplay /dev/vg01/lv01
  --- Logical volume ---
  LV Path                /dev/vg01/lv01         (1)
  LV Name                lv01
  VG Name                vg01                   (2)
  LV UUID                FVmNel-u25R-dt3p-C5L6-VP2w-QRNP-scqrbq
  LV Write Access        read/write
  LV Creation host, time servera.lab.example.com, 2022-04-07 10:45:34 -0400
  LV Status              available
  # open                 1
  LV Size                300.00 MiB             (3)
  Current LE             75                     (4)
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:0
```

1. Mit `LV Path` wird der Gerätename des LV angezeigt.
2. Mit `VG Name` wird die VG angezeigt, die zum Erstellen dieses LV verwendet wurde.
3. Mit `LV Size` wird die Gesamtgröße des LV angezeigt. Überprüfen Sie mit den Dateisystemtools den freien und belegten Speicherplatz für das LV.
4. `Current LE` zeigt die Anzahl der logischen Extents an, die von diesem LV genutzt werden.

14.2.8. Erweitern und Reduzieren von LVM-Storage

- Nach der Erstellung eines logischen Volumens kann dieses vergrössert werden um das Dateisystem zu erweitern
- Eventuell muss das PV oder die VG erweitert werden, um die Storage-Kapazität des LV zu erhöhen



14.2.8.1. Vergrössern einer Volume-Gruppe

- Eventuell muss mehr Festplattenspeicher hinzugefügt werden, um ein VG zu vergrössern
- physische Gerät vorbereiten und physische Volume erstellen (falls nicht vorhanden)

```sh
[root@host ~]# parted /dev/vdb mkpart primary 1072MiB 1648MiB
...output omitted...
[root@host ~]# parted /dev/vdb set 3 lvm on
...output omitted...
[root@host ~]# udevadm settle
[root@host ~]# pvcreate /dev/vdb3
  Physical volume "/dev/vdb3" successfully created.
```

- `vgextend`: der VG das neue PV hinzufügen

```sh
[root@host ~]# vgextend vg01 /dev/vdb3
  Volume group "vg01" successfully extended
```

Mit diesem Befehl wird die VG vg01 um die Grösse des PV /dev/vdb3 erweitert.

14.2.8.2. Vergrössern eines logischen Volumes

- logische Volumes können ohne Ausfallzeiten vergrössert werden
- dem LV in der VG freie physische Extents hinzufügen, um die Kapazität zur Erweiterung des Dateisystems des LV zu erhöhen

```sh
[root@host ~]# lvextend -L +500M /dev/vg01/lv01
  Size of logical volume vg01/lv01 changed from 300.00 MiB (75 extents) to 800.00 MiB
(200 extents).
  Logical volume vg01/lv01 successfully resized.
```

Dieser Befehl erhöht die Kapazität des logischen Volumes lv01 um 500 MiB. Das Pluszeichen (+) vor der Grösse bedeutet, dass dieser Wert der vorhandenen Grösse hinzugefügt wird. Andernfalls (also ohne das Plus-Zeichen) gibt der Wert die endgültige Grösse des LV an.

14.2.8.3. Erweitern eines XFS-Dateisystems auf die Grösse des logischen Volumes

`xfs_growfs` : Dateisystem erweitern zur Belegung des erweiterten LV

```sh
[root@host ~]# xfs_growfs /mnt/data/
...output omitted...
data blocks changed from 76800 to 204800
```

14.2.8.4. Erweitern eines EXT4-Dateisystems auf die Grösse des logischen Volumes

`resize2fs` : Dateisystem zur Belegung des neuen erweiterten LV erweitern (Dateisystem kann während der Grössenänderung weiterhin verwendet werden)

```sh
[root@host ~]# resize2fs /dev/vg01/lv01
resize2fs 1.46.5 (30-Dec-2021)
Resizing the filesystem on /dev/vg01/lv01 to 256000 (4k) blocks.
The filesystem on /dev/vg01/lv01 is now 256000 (4k) blocks long.
```

14.2.8.5. `xfs_growfs` vs. `resize2fs`

- Hauptunterschied: Argument welches übergeben wird
  - `xfs_growfs` übernimmt den Mount-punkt
  - `resize2fs` verwendet den LV-Namen
- Online-Grössenänderung: `xfs_growfs` , `resize2fs`
- Offline-Grössenänderung: `resize2fs`
- ext4-Dateisystem kann nach oben oder unten angepasst werden
- XFS-Dateisystem kann nur erhöht werden

14.2.9. Reduzieren des Volume-Gruppen-Storage

- nicht verwendete PVs werden aus der VG entfernt
- `pvmove` : Daten von Extents auf einem PV zu Extents auf einem anderen PV mit genügend freien Extents in derselben VG verschieben
- Während der Verkleinerung kann das LV derselben VG weiterhin verwendet werden.
- Option `-A` : Metadaten der VG nach einer Änderung automatisch sichern (verwendet den Befehl `vgcfgbackup` zur automatischen Sicherung der Metadaten)
- `vgreduce` : PV aus VG entfernen

```sh
[root@host ~]# pvmove /dev/vdb3
```

```sh
[root@host ~]# vgreduce vg01 /dev/vdb3
```

Hinweis: Die GFS2- und XFS-Dateisysteme unterstützen keine Verkleinerung, sodass die Grösse eines LV nicht verringert werden kann.

14.2.10. Entfernen von LVM-Storage

mit `lvremove, vgremove, pvremove` kann nicht mehr benötigte LVM-Komponente entfernt werden

- Vorbereiten des Dateisystems
- Entfernen des logischen Volumes
- Entfernen der Volume-Gruppe
- Entfernen der physischen Volumes

```sh
[root@host ~]# umount /mnt/data
```

```sh
[root@host ~]# lvremove /dev/vg01/lv01
Do you really want to remove active logical volume vg01/lv01? [y/n]: y
  Logical volume "lv01" successfully removed.
```

```sh
[root@host ~]# vgremove vg01
  Volume group "vg01" successfully removed
```

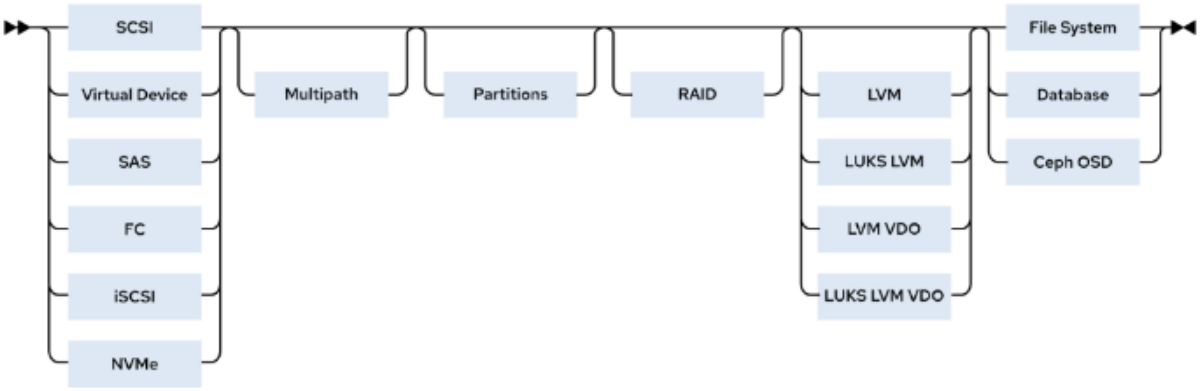
```sh
[root@host ~]# pvremove /dev/vdb1 /dev/vdb2
  Labels on physical volume "/dev/vdb1" successfully wiped.
  Labels on physical volume "/dev/vdb2" successfully wiped.
```

Dieser Befehl löscht die PV-Metadaten von der Partition (oder von dem Datenträger). Die Partition steht nun für die Neuzuweisung oder Neuformatierung zur Verfügung.

14.3. Verwalten von mehrschichtigem Storage

14.3.1. Storage-Stack

- Storage in Red Hat Enterprise Linux (RHEL) besteht aus mehreren Schichten
- Eine Schicht kann durch folgendes bestimmt werden:
  - Treiber,
  - Manger und/oder
  - Dienstprogramme



14.3.2. Block-Device

- Datenspeicher, der Daten in festen Blöcken überträgt (im Gegensatz zu Zeichengeräten, die Daten stromweise übertragen)
- Block-Device stehen am Ende des Storage Stacks (die tatsächliche Hardware oder das virtuelle Laufwerk)
- Bieten ein stabiles, konsistentes Geräteprotokoll, mit dem praktisch jedes Block-Device transparent in eine RHEL-Storage-Konfiguration einbezogen werden kann.
- Auf die meisten Block-Devices wird heute über den RHEL-SCSI-Gerätetreiber zugegriffen.
- Werden als SCSI-Geräte angezeigt, einschliesslich älterer ATA-Festplatten, Solid-State-Geräte und gängiger Enterprise-HBAs (Host Bus Adapter).
- RHEL unterstützt auch iSCSI, Fibre Channel over Ethernet (FCoE), Virtual Machine Driver (virtio), Serial-Attached Storage (SAS), Non-Volatile Memory Express (NVMe) und andere.
  - Ein iSCSI-Ziel kann (a) ein dediziertes physisches Gerät in einem Netzwerk oder ein per Software konfiguriertes logisches iSCSI-Gerät auf einem vernetzten Storage-Server sein. Das Ziel (b) ist der Portalendpunkt in einer SCSI-Protokollbuskommunikation, um auf den Storage als Logical Unit Numbers (LUNs) zuzugreifen.
  - Das FCoE-Protokoll überträgt Fibre Channel-Frames über Ethernet-Netzwerke. ⟶ In der Regel verfügen Rechenzentren über dedizierte LAN- und SAN-Verkabelungen (Storage Area Network), die für den jeweiligen Datenverkehr eindeutig konfiguriert sind. Mit FCoE können beide Arten von Datenverkehr in einer grösseren, konvergenten Ethernet-Netzwerkarchitektur kombiniert werden. Zu den Vorteilen von FCoE zählen niedrigere Hardware- und Energiekosten.

14.3.2.1. Zusammenfassung

Block-Devices sind die grundlegenden Bausteine für Speicher in RHEL. Das Betriebssystem abstrahiert die Hardware, indem es fast alles (von der lokalen SSD bis zum komplexen Netzwerkspeicher via iSCSI oder FCoE) über einen standardisierten SCSI-Treiber anspricht. Dies macht die Verwaltung von Speicher effizienter und kostengünstiger, da verschiedene Technologien einheitlich genutzt werden können.

14.3.3. Multipath

Ziel: Ausfallsicherheit und Leistung beim Zugriff auf die Datenspeicher erhöhen

Grundlage: setzt mehrere gleichzeitige Verbindungswege -> Redundanz (failover) und Leistungssteigerung (Load Balancing)

- Pfad ist eine Verbindung zwischen einem Server und dem Storage (der darunter liegt)
- Device Mapper Multipath (dm-multiplath) -> natives RHEL-Multipath Tool zur Konfiguration redundanter I/O Pfade ein einem einzelnen logischen Geräte mit Pfad-Aggregation
  - Beispiel: Im Rechenzentrum möchte ich zB sicherstellen, dass auch beim Ausfall eine echten, physikalischen Verbindung zum Storage-System eine alternative Verbindung genutzt und bereitgestellt werden kann. ⟶ alternativer Pfad zum Storage.
- Ein mit der Gerätezuweisung (dm) erstelltes logisches Gerät wird unter `/dev/mapper/` als eindeutiges Block- Device für jede an das System angeschlossene LUN angezeigt.
- Storage-Multipath-Redundanz kann auch durch Netzwerkbündelung implementiert werden, wenn der Storage, wie z. B. iSCSI und FCoE, Netzwerkkabel verwendet.

14.3.4. Partitionen

- Unterteilung der Block-Geräte in kleinere Einheiten
- ein Block-Device kann weiter in Partitionen unterteilt werden
- Partitionen können die gesamte Block-Device-Grösse beanspruchen oder das Block-Device aufteilen, um so mehrere Partitionen zu erstellen.
- Diese Partitionen können dann verwendet werden, um ein Dateisystem oder LVM-Geräte zu erstellen, oder sie können direkt für Datenbankstrukturen oder anderen unformatierten Storage verwendet werden.

14.3.5. RAID

- Redundant Array of Inexpensive Disks
- Bietet Datenredundanz und Leistungssteigerung durch Spiegelung oder Stripping
- Technologie zur Storage-Virtualisierung
- Beispiele:
  - RAID 0: Festplatten-Striping
  - RAID 1: Festplattenspiegelung
  - RAID 1 + 0: Festplattenspiegelung und Striping
  - RAID 6: Festplatten-Striping mit doppelter Parität
- LVM unterstützt RAID-level, 0, 1, 4, 5, 6 und 10
- Logische RAID-Volumes, die von LVM erstellt und verwaltet werden, nutzen die Kernel-Treiber „Multiple Devices“ (mdadm). Wenn LVM nicht verwendet wird, stellt Device Mapper RAID (dm-raid) eine Gerätezuweisungsschnittstelle für die mdadm-Kernel-Treiber bereit.

14.3.6. Logical Volume Manager

- ermöglichst die dynamische Grössenänderung von Volumes ohne Systemstopp
- Unterstütz Snapshots für Sicherungskopien und RAID-Level
- Integrierte Verschlüsselung (LUKS) und Datenoptimierung (VDO)

14.3.6.1. LUKS

Vorteil gegenüber dateisystem- oder dateibasierten Verschlüsselung: ein LUKS-Verschlüsseltes Gerät ermöglicht keine öffentliche Sicherbarkeit bzw. keinen Zugriff auf die Dateisystemstruktur (ein physisches Gerät bleibt auch dann sicher, wenn es von einem Computer entfernt wird)

14.3.6.2. Virtual Data Optimizer (VDO)

- bietet **Deduplizierung** (Vermeidung doppelter Blöcke), **Komprimierung** und **Thin Provisioning**
- Ziel ist die effiziente Nutzung des verfügbaren physischen Speichers

14.3.7. Dateisystem oder andere Verwendung

Die oberste Schicht des Storage-Stacks wird normalerweise durch ein Dateisystem gebildet. Alternativ kann dieser Platz als unformatierter Speicher (Raw Storage) für Datenbanken oder spezielle Anwendungen genutzt werden.

14.3.7.1. Dateisysteme in RHEL

- RHEL unterstützt verschiedene Dateisystemtypen, spricht jedoch für die meisten aktuellen Anwendungsfälle eine Empfehlung für **XFS** aus.
- XFS ist zwingend erforderlich, wenn Red Hat Ceph Storage (via LVM-Implementierung) oder das Stratis-Storage-Tool eingesetzt werden.

14.3.7.2. Nutzung durch Datenbanken

Datenbanken nutzen Speicher auf unterschiedliche Weise:

- **Kleine Datenbanken:** Speichern Strukturen oft in regulären Dateien innerhalb eines Dateisystems. Dies kann jedoch zu Einschränkungen bei der Skalierung führen (wegen Overhead oder Zugriffsbeschränkungen).
- **Grosse Datenbanken:** Bevorzugen oft unformatierten Speicher, um Dateisystem-Caching zu umgehen und eigene Caching-Mechanismen zu nutzen.
- **Logische Volumes:** Diese eignen sich sowohl für Datenbanken als auch für Szenarien mit unformatiertem Speicher.

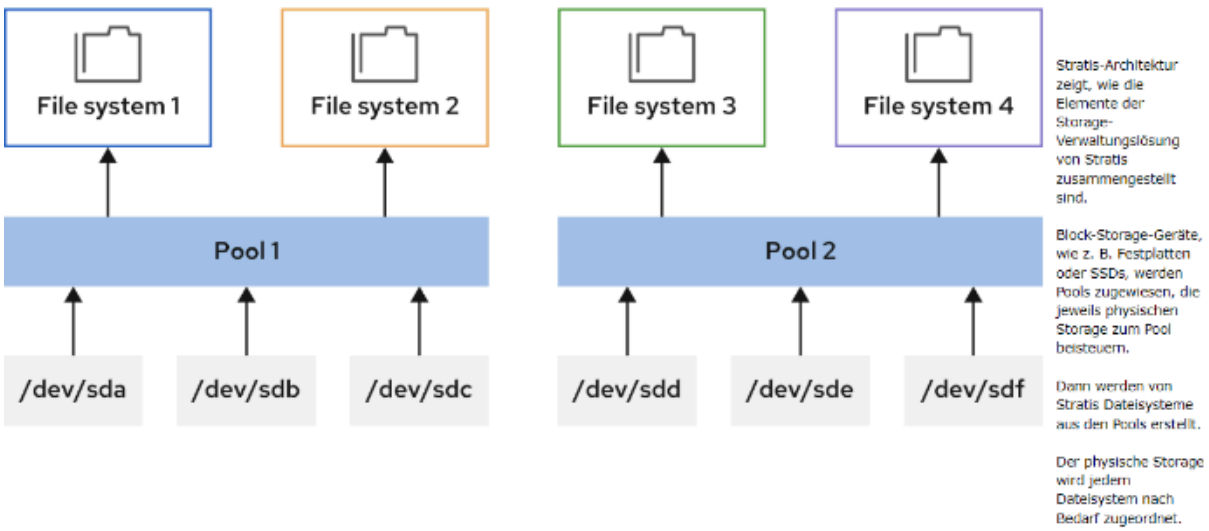
14.3.7.3. Red Hat Ceph Storage

Ceph ist eine umfassende Speicherlösung mit besonderen Merkmalen bei der Datenverwaltung:

- **Speicherverwaltung:** Ceph bevorzugt unformatierte Geräte, um eigene Metadatenstrukturen für Ceph Object Storage Devices (OSDs) zu erstellen. In aktuellen Versionen wird Ceph LVM genutzt, um Disk-Geräte als OSDs zu initialisieren.
- **CephFS:** Die Lösung verfügt über ein eigenes Dateisystem namens Ceph-Filesystem.
- **Netzwerk und Sicherheit:** Daten können über verschiedene Komponenten im Netzwerk verteilt abgelegt werden. Zudem ist eine mehrfache Sicherung der Daten in physisch unterschiedlichen Speicherbereichen möglich.

14.3.8. Stratis

- Tool für lokale Storage-Verwaltung
- gestaltet die LVM-Konfiguration einfacher
- Stratis nutzt aktuell LVM + Thin Provisioning und XFS als Haupt-Schichten bei der Bereitstellung von Storage
- wird als Service ausgeführt
  - verwaltet Pools physischer Speichergeräte
  - erstellt und verwaltet transparent Volumes für die neu erstellen Dateisystme
- dynamische Zuweisung (statt sofort bei der Erstellung)
  - Aus diesem Grund scheint das Dateisystem z. B. 1 TiB gross zu sein, hat jedoch möglicherweise nur 100 GiB des tatsächlichen Speichers aus dem Pool zugewiesen.



14.3.8.1. Administrationsmethoden von Stratis

- Pakete `stratis-cli` und `stratisd` installieren
  - `stratis-cli` : stellt Befehl `stratis` zur Verfügung, der Neukonfigurationsanforderungen an den System-Daemon `stratisd` sendet

  - `stratisd`: stellt Service `stratisd` bereit, der die Neukonfigurationsanforderungen verarbeitet und Block-Devices, Pools, und Dateisysteme, die Stratis verwendet, verwaltet und überwacht
- Stratis-Administration ist in der RHEL-Webkonsole enthalten
- von Stratis erstellte Dateisysteme sollen nur mit Stratis-Tools und -Befehlen neu konfiguriert werden
  - Stratis verwendet Metadaten
  - andere Befehle können dazu führen, dass Metadaten überschrieben werden

14.3.8.2. Installieren und Aktivieren von Stratis

1. Pakete `stratis-cli` und `stratisd` installieren: `dnf install stratis-cli stratisd`
2. Service Starten und aktvieren: `systemctl enable --now stratisd`
3. Stratis-Pools erstellen: `stratis pool create`

```sh
[root@host ~]# stratis pool create pool1 /dev/vdb
[root@host ~]# stratis pool list
Name                    Total Physical   Properties              UUID
pool1   5 GiB / 37.63 MiB / 4.96 GiB      ~Ca,~Cr    11f6f3c5-5...
```

4. einem Pool zusätzliche Block-Devices hinzufügen: `stratis pool add-data` (Überprüfung: `stratis blockdev list`)

```sh
[root@host ~]# stratis pool add-data pool1 /dev/vdc
[root@host ~]# stratis blockdev list pool1
Pool Name   Device Node   Physical Size   Tier
pool1       /dev/vdb              5 GiB   Data
pool1       /dev/vdc              5 GiB   Data
```

5. Dateisystem aus einem Pool erstellen: `stratis filesystem create`

```sh
[root@host ~]# stratis filesystem create pool1 fs1
[root@host ~]# stratis filesystem list
Pool Name   Name   Used      Created             Device                  UUID
pool1       fs1    546 MiB   Apr 08 2022 04:05   /dev/stratis/pool1/fs1  c7b5...
```

6. Snapshot erstellen `stratis filesystem snapshot`
   - Stratis weist den Speicherplatz dynamisch zu und verwendet anfänglich 560 MB für die Speicherung des Journals des Dateisystems

```sh
[root@host ~]# stratis filesystem snapshot pool1 fs1 snapshot1
```

14.3.8.3. Dauerhaftes Mounten von Stratis-Dateisystemen

Datei `/etc/fstab` bearbeiten um ein Stratis-Dateisystem dauerhaft zu mounten

Folien:

```txt
UUID=c7b57190-8fba-463e-8ec8-29c80703d45e /dir1 xfs defaults,x-
systemd.requires=stratisd.service 0 0
```

- Die Mount-Option `x-systemd.requires=stratisd.service` verzögert das Einhängen (Mounten) des Dateisystems, bis der Daemon systemd während des Bootvorgangs den Service stratisd gestartet hat.

Übung:

```txt
/dev/stratis/mypool/myfilesystem /mnt/mystratis xfs defaults,x-systemd.requires=stratis-
fstab-setup@pool-uuid.service,x-systemd.after=stratis-fstab-setup@pool-uuid.service 0 0
```

- `x-systemd.requires=stratis-fstab-setup@pool-uuid.service` : Stellt sicher, dass der Stratis-Setup-Dienst vor dem Mounten startet.
- `x-systemd.after=stratis-fstab-setup@pool-uuid.service` : Legt die Reihenfolge fest, damit die Voraussetzungen für den Mount erfüllt sind.

Beispieleintrag in `/etc/fstab` finden:

```sh
[root@host ~]# lsblk --output=UUID /dev/stratis/pool1/fs1
UUID
c7b57190-8fba-463e-8ec8-29c80703d45e
```

- **Problem mit `df` :** Der Befehl `df` sollte nicht zur Abfrage von Stratis-Dateisystemen genutzt werden.
- **Falsche Anzeige:** `df` meldet pauschal eine Grösse von 1 TiB, unabhängig von der tatsächlichen Zuweisung.
- **Thin Provisioning:** Dateisysteme werden virtuell bereitgestellt; der physische Speicher im Pool kann kleiner sein als die Summe der (scheinbaren) Dateisystemgrössen.
- **Gefahr von Schreibfehlern:** Ein Pool kann vollaufen, auch wenn `df` noch freien Speicher anzeigt. Dies führt zu fehlgeschlagenen Schreibvorgängen.
- **Lösung:** Zur genauen Überwachung des verfügbaren Speicherplatzes immer den Befehl **`stratis pool list`** verwenden.

14.4. LVM vs. Stratis

Merkmal	LVM (Logical Volume Manager)	Stratis
Physische Schicht	Physische Volumes (PVs)	Blockgeräte
Pool-Konzept	Volume Groups (VGs) aus PVs	Stratis Pools aus Blockgeräten
Logische Volumes	Logische Volumes (LVs)	Stratis-Dateisysteme
Dateisystemtyp	Flexibel (ext4, xfs etc.)	Standardmässig XFS
Speicherverwaltung	Manuelle Zuweisung von Extents	Automatische Verwaltung
Snapshots	Unterstützt, manuell zu verwalten	Unterstützt, benutzerfreundlich
Metadaten	Benötigt eigene Metadatenverwaltung	Versteckt Komplexität, nutzt device-mapper
Erweiterbarkeit	Manuell durch PV-Erweiterung	Einfaches Hinzufügen von Blockgeräten
Flexibilität bei Typen	Hohe Flexibilität mit verschiedenen Typen	Weniger abweichende Typen, fokussiert auf XFS
Abstraktion	Weniger, detaillierte Kontrolle	Mehr Abstraktion für Einfachheit

15. Befehlsübersicht

Befehl	Beschreibung
`whoami`	Zeigt aktuellen Benutzer an
`date`	Gibt aktuelles Datum und Uhrzeit aus
`history`	Liste der zuletzt ausgeführten Befehle
`exit`	Aktuelle Shell-Sitzung beenden
`shutdown`	System herunterfahren / rebooten
`systemctl`	Verwalten von Systemd-Units/Diensten
`pwd`	Aktuelles Arbeitsverzeichnis anzeigen
`ls`	Dateien/Verzeichnisse auflisten
`cd`	Verzeichnis wechseln
`cat`	Datei-Inhalt ausgeben
`less`	Textdateien seitenweise betrachten (besser als `cat` für grosse Dateien)
`head`	Erste Zeilen einer Datei anzeigen
`tail`	Letzte Zeilen einer Datei anzeigen
`wc`	Zeilen, Wörter, Zeichen zählen
`file`	Dateityp ermitteln
`mkdir`	Erzeugt ein Verzeichnis
`cp`	Datei oder Verzeichnis kopieren
`mv`	Verzeichnis/Datei verschieben oder umbenennen
`rm`	Datei oder Verzeichnis löschen
`rmdir`	Leeres Verzeichnis löschen
`touch`	Leere Datei erstellen oder Zeitstempel ändern
`ln`	Links erstellen (Soft- und Hardlinks)
`chown`	Eigentümer von Dateien/Verzeichnissen ändern
`chmod`	Dateiberechtigungen ändern
`su`	Benutzer wechseln / Root-Shell starten
`sudo`	Befehl mit Root-Rechten ausführen
`passwd`	Passwort eines Benutzers ändern
`useradd`	Benutzer erstellen
`usermod`	Benutzer-Einstellungen ändern
`userdel`	Benutzer löschen
`getent`	Einträge aus Systemdatenbanken lesen
`id`	User- und Gruppen-IDs anzeigen
`ps`	Laufende Prozesse anzeigen
`top`	Interaktive Prozessüberwachung
`kill`	Signal an Prozess senden (beenden)
`killall`	Prozesse nach Name beenden
`pkill`	Prozesse nach Suchmuster beenden
`journalctl`	Zentrale Abfrage des Systemd-Journals (Logs)
`tar`	Archivieren von Dateien

Befehl	Beschreibung
`scp`	Dateien sicher über SSH kopieren
`sftp`	Sicherer FTP-Modus über SSH
`rsync`	Dateisynchronisation
`rpm`	RPM-Paketmanager
`dnf`	Softwarepakete verwalten (DNF)
`rpm2cpio`	Extrahieren von RPM-Dateien
`ip`	Netzwerkeinstellungen (Link/Addr/Route)
`nmcli`	NetworkManager Konfiguration
`ping`	Netzwerkverbindung prüfen
`tracepath`	Paketpfad verfolgen
`ss`	Aktive Socket-Verbindungen anzeigen
`vim`	Texteditor
`grep`	Muster in Text suchen
`apropos`	Man-Pages durchsuchen
`man`	Handbuchseite eines Befehls anzeigen
`echo`	Text oder Variablen ausgeben
`env`	Umgebungsvariablen anzeigen
`set`	Systemvariablen anzeigen
`unset`	Variablen löschen
`test`	Bedingungen prüfen
`at`	Einmalige Jobplanung
`crontab`	Wiederkehrende Jobs verwalten
`lsblk`	Zeigt Blockgeräte (Festplatten/Partitionen) in Baumstruktur an
`df`	Anzeige des freien Speicherplatzes auf Dateisystemen
`du`	Speicherverbrauch von Verzeichnissen schätzen
`parted`	neue Partition erstellen
`mount` / `umount`	Dateisysteme einhängen oder trennen
`stratis`	Stratis-Storage verwalten