Grundlagen der Programmierung

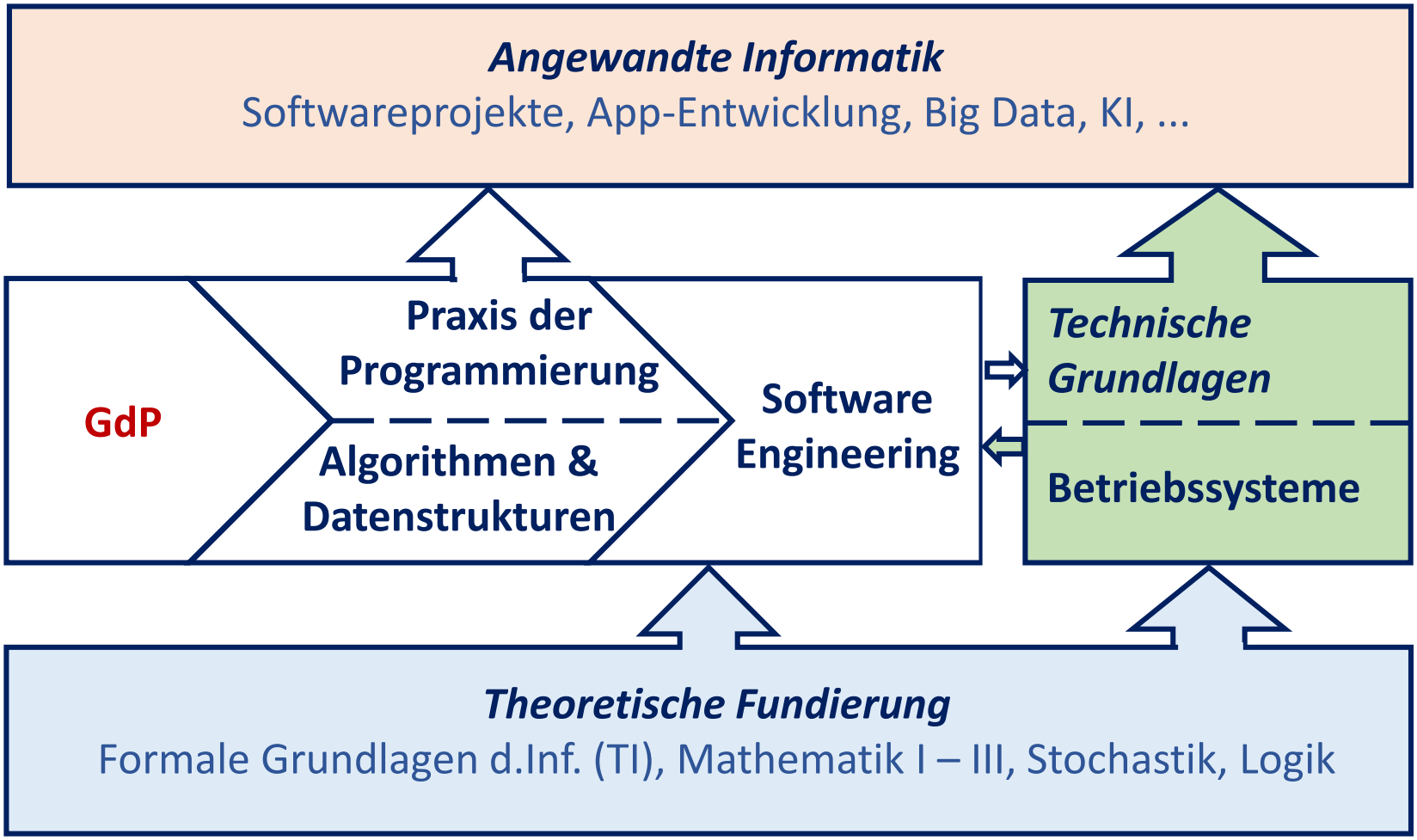
Einführung:

Zielstellung ♦ Organisatorisches

Ziel des Kurses GdP

- **Algorithmisches Denken**
  - Vom Problem zum Algorithmus
  - Güte und Grenzen von Algorithmen
- **Programmieren**
  - Vom Algorithmus zum Programm
  - Programmieren mit **Python**
  - Programmierparadigmen
  - Interpreter, Compiler, Assembler
- **Betriebssystem UNIX/Linux**

GdP im Curriculum



O-Notation

Sortieralgorithmen

Qualität von Algorithmen

Kürzeste Wege

Unentscheidbare Probleme

Diagonalisierung

Grenzen von Algorithmen

Pseudocode

Brute-Force

Assembler

Compiler

Interpreter

Stack und Queue

Iteration

Kontrollstrukturen

Prozesse

Algorithmisches Denken

Betriebssystem

Rekursion

Paradigmen

Graphen

Prozedurale Programmierung

Funktionale Programmierung

Lambda-Ausdrücke

Iterative Programmierung

Logische Programmierung

Currying

Deklarative Programmierung

Objektorientierte Programmierung

Asymptotische Komplexität

Ein paar Fragestellungen

- Wie geht man beim Entwurf eines Algorithmus vor?
- Mit welchen Konzepten formuliert man Algorithmen?
- Wie werden diese Konzepte beim Programmieren umgesetzt?
- Wie vergleicht man verschiedene Algorithmen, die dasselbe Problem lösen (*objektiv/mathematisch*)?
- Gibt es Probleme, die man mit Hilfe von Algorithmen überhaupt nicht lösen kann?
  - Kann man das beweisen?
  - Trifft man oft auf solche Probleme?
  - Und was macht man dann?

Vorlesung – Übung – Rechnerübung

- **Vorlesung**
  - Vermittlung der **Konzepte**
  - Einführung in **Python**
  - Voraussetzung für erfolgreiche Teilnahme an den Übungen, Rechnerübungen und der Prüfung
- **Übung**
  - Vertiefung des Vorlesungsstoffs durch weitere Beispiele
  - S*elbstständiges* Finden von Lösungsansätzen
- **Rechnerübung**
  - Umsetzung der Konzepte
  - Praktischer Umgang mit **UNIX/Linux** und **Python**

Was ist (noch) anders als in der Schule?

- sehr große Stoffdichte
- keine laufende Leistungsüberprüfung
- Sie müssen selbst dafür sorgen, dass Sie mitkommen!
  - Besuch der Lehrveranstaltungen!!!
  - Nacharbeiten der Vorlesung
  - Vorbereiten der Übungen (Aufgaben/ Ihre Fragen)
  - Bilden Sie Lerngruppen!
- Bleiben Sie am Ball!
  Denn der Stoff baut aufeinander auf.

Ablauf der Vorlesungen

- **Folien**

  enthalten alle Begriffe, Definitionen, Aussagen und einige Beispiele und Erklärungen

- Viele Beispiele, Überlegungen und Erklärungen werden schrittweise entwickelt.

  Schreiben Sie mit! Das wird Ihnen helfen!

Ablauf der Rechnerübungen

- selbstständige und angeleitete Arbeit im Computerlabor (2.70.0.01 und 2.70.0.05)
- Account beantragen: https://www.uni-potsdam.de/de/cs/ifi/services/accountverwaltung
- Übungsblätter
  - mit ausführlichen Aufgabenstellungen
  - z.T. mit Anleitungen und Hinweisen zur Selbstkontrolle
- Betreuung durch Tutoren
  - Hilfestellung bei Verständnisfragen oder technischen Problemen
  - Führen durch Aufgaben mit größerer Komplexität
  - Vergleich von Lösungen / Erklären von Lösungsschritten

Ablauf der Übungen

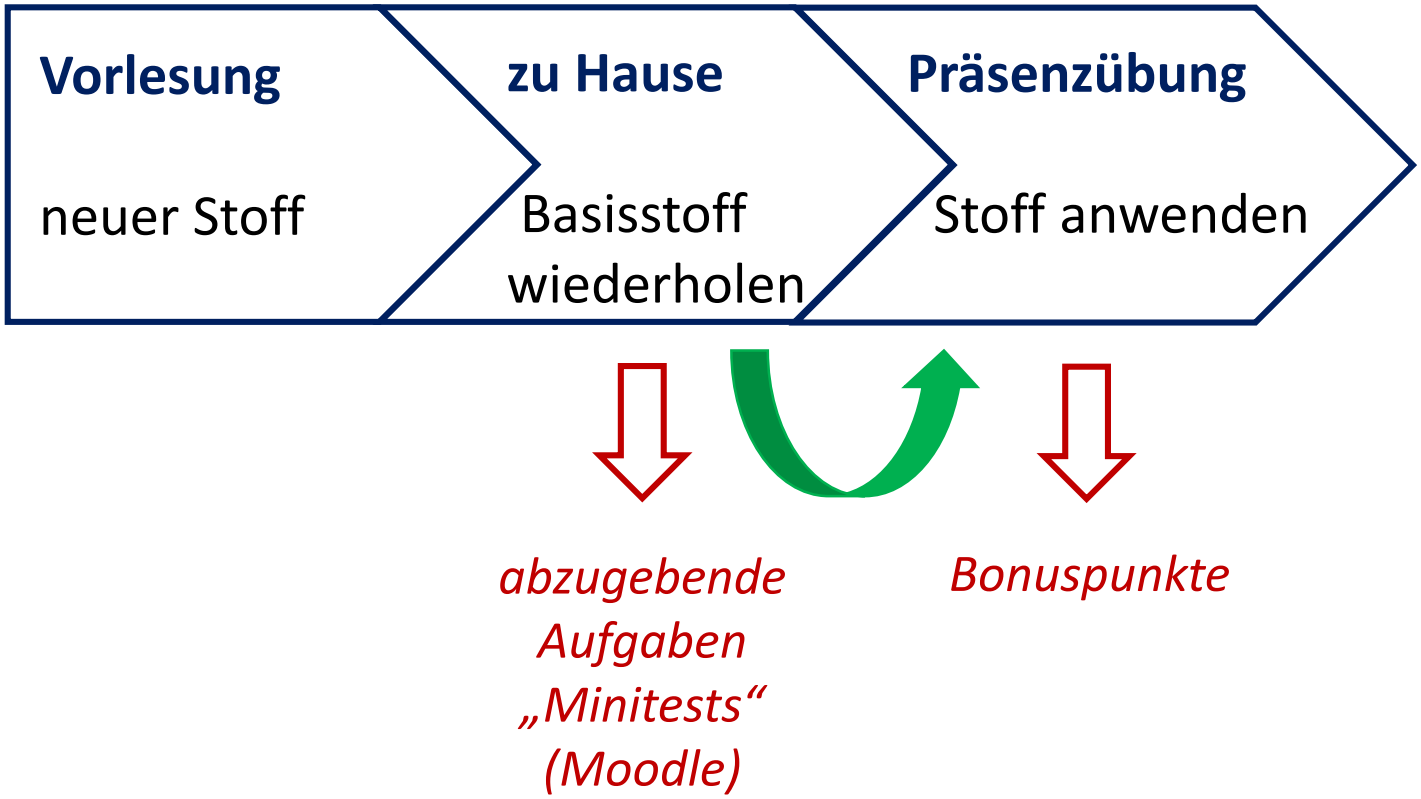
- Übungsaufgaben, die Sie vorher zu Hause lösen
- Vorstellung Ihrer Lösungen zu den Übungsaufgaben

  → ***Bonuspunkte*** *für die Klausur durch das Vorstellen von Lösungsschritten*
- *und das Wichtigste ...*
  Diskussion Ihrer **Fragen** zum Vorlesungsstoff

Vorbereitung auf die Übungen

- Durchgehen der Vorlesung
- Fragen notieren!!! → *in den Übungen diskutieren*
- Abgabe von ausgewählten Aufgaben *via Moodle* vor den zugehörigen Übungen (*Basiswissen*)
  - ab 4. LV-Woche (06.11.-10.11.23),
    der erste noch ohne Bewertung (als „Generalprobe“)
  - zunächst Übungsaufgaben mit Feedback, wiederholbar
  - dann ähnliche Aufgaben als Test (nicht wiederholbar)
  - ***Voraussetzung für die Zulassung zur Klausur:***
    jeweils mindestens 60 % korrekt gelöst; bis zu 3 „Patzer“
- Vorbereitung der Aufgaben für die Präsenzübung

Lernprozess



Leistungserfassung

- **Moodle-Tests zum Vorlesungsstoff (PNL zur Vorlesung/Übung)**
  - Voraussetzung für die Zulassung zur Modulprüfung
- **Test UNIX/Linux (PNL zur Rechnerübung)**
  in der Rechnerübung während der 4. LV-Woche (06.-10.11.23), im Rechnerpool zu absolvieren
  - muss bestanden werden (50%), um das Modul abzuschließen
- **Klausur (Modulprüfung)**
  120 Minuten, ohne Unterlagen
  - Bestehen durch mindestens 50% der Punkte
- **Bonuspunkte** aus den Übungen zählen für die Klausur.
  Der Kurs muss aber ohne diese bestanden werden!

Die ersten fünf Wochen

Die Übungen in den ersten drei Wochen werden als Rechnerübungen genutzt.

	Vorlesung	Übung	Rechnerübung
1	Org., Grundbegriffe, UNIX, UNIX-Prozesse	**RÜ**	**RÜ**
2	UNIX-Dateisystem	**RÜ**	**RÜ**
3	**Math. Grundlagen**	**RÜ**	**RÜ**
4	Algorithmus, Pseudocode	---	**TEST**
5	Modelle, Graphen, Brute Force	**Math. Grundlagen**	**Python**

Informationszugang

Moodle:

- https://moodle2.uni-potsdam.de/ (Kurs: GdP_23/24)
- **Gruppenzuordnung** (nur Moodle ist verbindlich!!!)
  - **Separate Gruppen für Lehramt**
- alle Lehrmaterialien (Folien, Übungsaufgaben)
- wöchentliche Tests und Übungen dazu (ab 4. LV-Woche)
- aktuelle Informationen
- zwei Foren

kein Einschreibeschlüssel

PULS

1. Belegen

- Vorlesung/Übung (genau eine Gruppe, *beliebig*)
- Rechnerübung (genau eine Gruppe, *beliebig*)
- Zulassungen erfolgen laufend
- Ohne Zulassung keine Prüfungsnebenleistungen!!!
- Modul INF-1010

2. Prüfungsanmeldung

- mindestens acht Werktage vor dem Prüfungstermin (also vor dem Klausurtermin)
- Sie werden von uns zugelassen.
- Ohne Anmeldung keine Klausurteilnahme!!!

Fragen ?!

Grundlagen der Programmierung

Einführung:
Grundbegriffe der Informatik

Grundbegriffe der Informatik

- Hardware, Software
- Algorithmus, Programm, Prozess
- Betriebssystem
- Computernetze

Hardware

- Gesamtheit aller technischen (mechanischen und elektronischen) Geräte eines informationsverarbeitenden Systems (eines Computers)

- als physische, materielle Teile *nicht* kopierbar oder (reversibel) veränderbar

Quelle: Wikipedia

Wesentliche Komponenten der Hardware (von-Neumann-Architektur)

→ Maschinenmodelle

- **Zentraleinheit**
  - **Prozessor**
    mit Registern, in denen Zahlenwerte gespeichert und vom Prozessor verändert werden können
  - **Arbeitsspeicher**/Primärspeicher
    zur Datenhaltung außerhalb der Register
  - **Busse** (Leitungen) zwischen beiden

- **Peripheriegeräte** (zur Ein-/Ausgabe von Daten)
  - Tastatur, Maus, Bildschirm, Drucker
  - **Sekundärspeicher** (Festplatte, CD-, DVD-, Flashlaufwerke)
  - Netzwerkinterface, ...
  - Verbindungsleitungen

Software

- Gesamtheit aller Programme, die auf einer Rechenanlage eingesetzt werden können, samt der dazugehörigen Daten
- können Änderung der Werte in den Registern und im Arbeitsspeicher der Zentraleinheit bewirken
- leicht veränderbar und kopierbar

Programm und Algorithmus

- Ein **Programm** ist ein Text (Code), der einen *Algorithmus* formuliert, so dass er maschinell ausgeführt werden kann.

- Ein **Algorithmus** ist eine Folge von Anweisungen, die *Eingabedaten* in *Ausgabedaten* überführt.

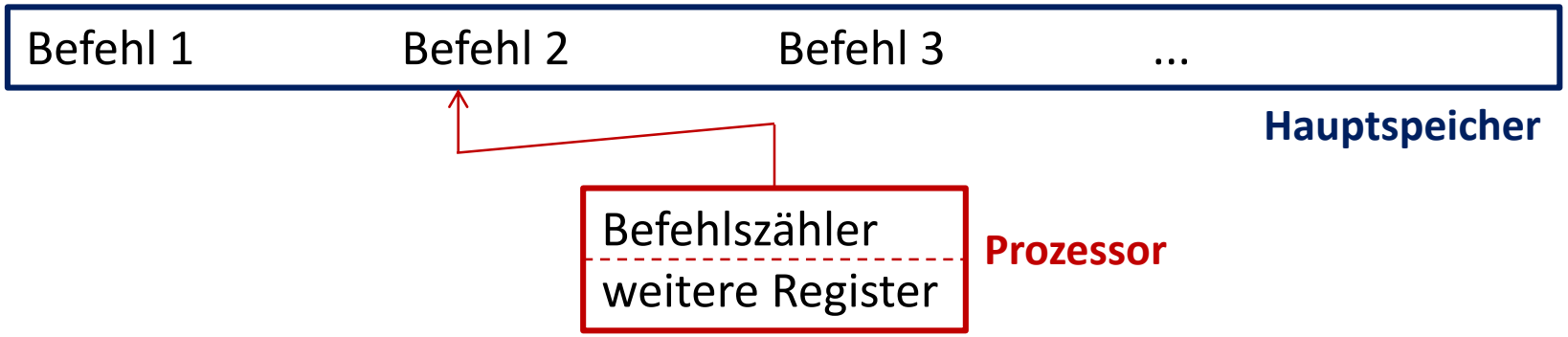
  Dabei muss für jede Eingabe eindeutig feststehen:

  - Welche Anweisung wird zuerst ausgeführt?
  - Welche Anweisung folgt auf eine gerade ausgeführte?
  - Wann ist der Algorithmus beendet?

Programm (2)

- Code wird in der Regel dauerhaft auf einem Sekundärspeicher gespeichert
- wird vor der Ausführung in den Primärspeicher geladen
- wird auf von-Neumann-Rechnern **sequenziell** ausgeführt

Der Prozessor enthält genau einen **Befehlszähler**, der stets die Adresse der nächsten auszuführenden Anweisung enthält.



Prozess (Task)

- ist ein gestartetes Programm, das gerade von einem Rechner ausgeführt wird

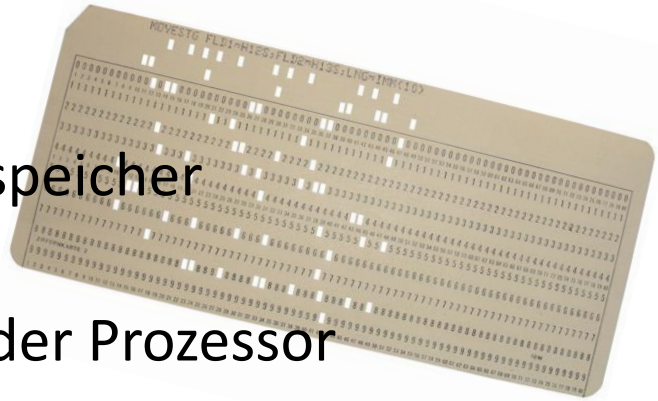
- *anschaulich:*
  Wanderung des Befehlszählers durch das Programm

- Auf von-Neumann-Rechnern kann zu jeder Zeit nur ein Prozess aktiv sein.

Betriebssystem

- Basissoftware, die den Betrieb eines Rechners ermöglicht
- übernimmt Aufgaben wie
  - Steuerung der Hardware
  - Speicherverwaltung
  - Verwaltung der Prozesse
  - Nutzerverwaltung
  - Kommunikation mit Benutzern (Kommandointerpretation)

Charakteristika moderner Betriebssysteme



- **Multiprogramming**
  Halten mehrerer Programme im Hauptspeicher
- **Multitasking**
  Verwaltung mehrerer Prozesse, denen der Prozessor abwechselnd zugeteilt wird
  → *Pseudo-Parallelität* durch *Time-Sharing/Scheduling*
- **Multiuser**
  Anmeldung mehrerer Benutzer gleichzeitig möglich
- **Multiprozessing**
  Zusammenarbeit mit mehreren Prozessoren
- **Netzwerkbetriebssysteme**
  erlauben Kommunikation mit anderen Rechnern

Lokale Netze

- **LAN – Local Area Network**
  meist in Gebäuden oder Gebäudekomplexen (*z.B.* Betrieben)
- *Hauptzweck:*
  gemeinsame Nutzung von Betriebsmitteln
  (Dateien, Prozessoren, Druckern, Diensten, ...)
- **Server** stellen die Betriebsmittel bereit
  - **Fileserver:** Dateien
  - **Applikationsserver:** Rechenleistung (starke Prozessoren)
  - **Mailserver/Webserver/... :** Dienste
  - **...**

Server und Clients

→ Grundlagen der Betriebssysteme und Rechnernetze

- **Server** sind Prozesse, die Betriebsmittel bereitstellen (die auf entfernten oder dem lokalen Rechner laufen)
- **Clients** sind Prozesse, die die Betriebsmittel nutzen

Server
- wartet auf Anfragen von Clients
- prüft Berechtigung
- stellt Betriebsmittel bereit und verwaltet Ressourcen

Client
- stellt bei Bedarf Anfragen an Server
- nutzt Betriebsmittel so, als wären sie lokal vorhanden

- Zusammenspiel von Clients und Servern in einem LAN erfordert spezielle Software (Netzwerkbetriebssystem)

Grundlagen der Programmierung

Einführung in UNIX/Linux

Historie ♦ Kommandozeile ♦ Prozesskonzept

Das Betriebssystem UNIX

- **UNIX:** eingetragenes Warenzeichen
  von *The Open Group* (`http://www.opengroup.org`)

  → Festlegung der **Spezifikation** eines Betriebssystems

- **UNIX-System:** ein Betriebssystem,
  das dieser Spezifikation entspricht

- Trennung von Warenzeichen und Code,
  so dass mehrere Implementierungen möglich sind

  → Es gibt kein einheitliches UNIX (Umfang und Interna).

UNIX-Versionen und -Derivate

- Zwei traditionelle Entwicklungslinien:
  1. **AT&T**: Version 1, ... Version 7, System III, ..., System V.4
  2. **Berkley**: 1BSD, ..., 4.4BSD (*Berkley Software Distribution*)
     → dominierend im Serverbereich
- weitere UNIX-Systeme für die Hardware verschiedener Hersteller:
  HP-UX, IBM AIX, Solaris, ...
- **POSIX:** Standard,
  der gewisse Interna von UNIX-Systemen vereinheitlicht
  (vor allem Systemaufrufe)

Linux

- Für PC-Prozessoren: verschiedene Linux-Distributionen
- Linux: Systemkern (ohne Oberflächen, Anwendungen etc.)
  (1991 vom finnischen Studenten Linus Torvalds programmiert)
- weitere Systemsoftware nötig, z.B. die des **GNU**-Projekts
- Linux-Kernel und GNU-Software sind Open Source
  (können von jedem frei angepasst und weiterentwickelt werden)
- Unabhängige Organisationen (Distributoren) sammeln und vertreiben Software für Linux.

  → In einer Linux-Distribution sind (fast) alle Programme vorhanden, die man zum komfortablen Rechnerbetrieb benötigt (*Editoren, Compiler, Office-Programme, ...*)

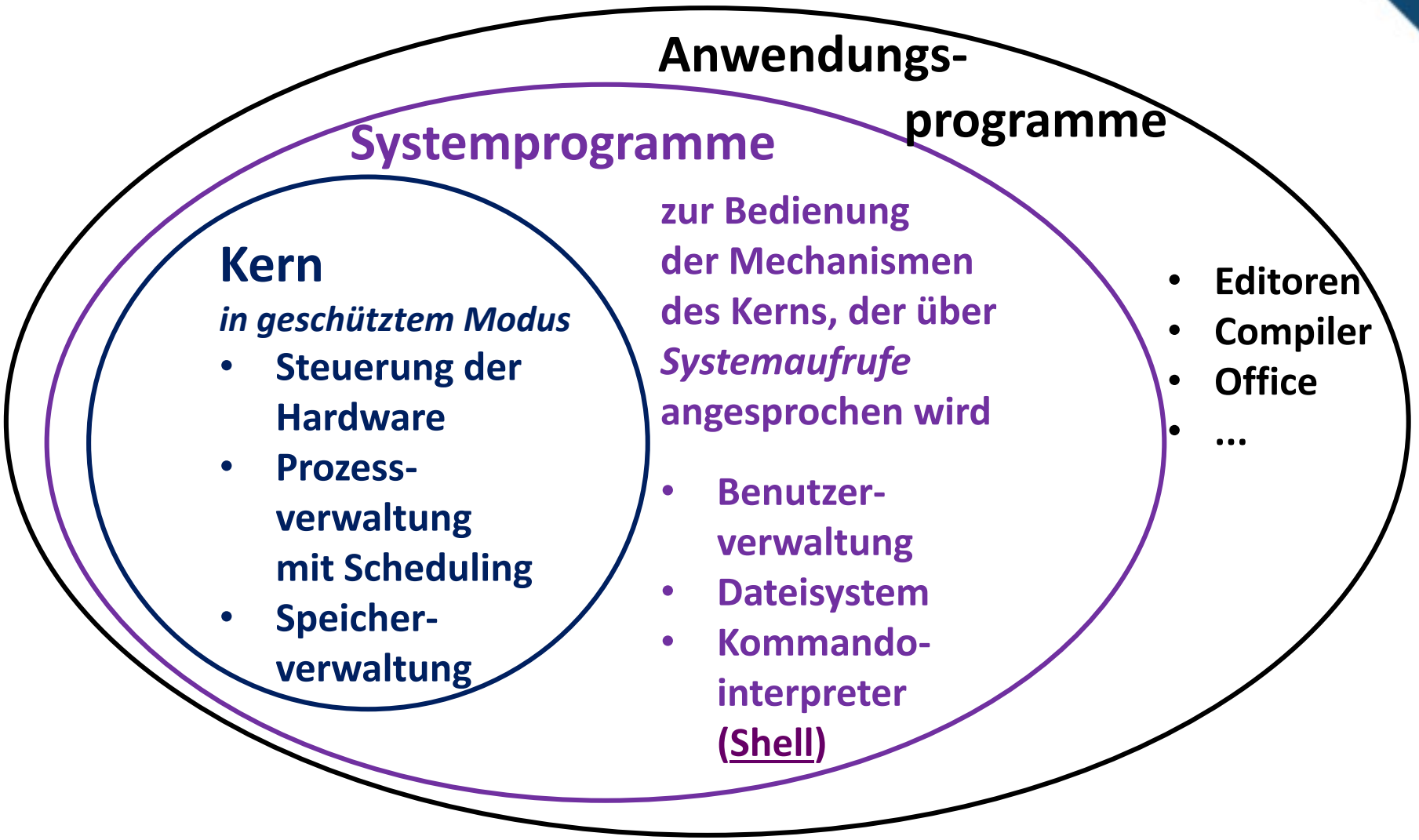
Linux-Distributionen

- www.suse.de
- www.ubuntu.com
- www.redhat.com
- www.debian.org – im Computerlabor verfügbar
- www.centos.org – **CentOS** (*Community Enterprise Operating System*)
- (www.knoppix.de) – **Live-System**
  - muss nicht installiert werden
  - alle Systemkomponenten werden in den Arbeitsspeicher geladen

Eigenschaften von UNIX/Linux

- Multiprozessor
- Multitasking
- Multiuser
- Netzwerkbetriebssystem
- sehr stabil, gute Portabilität, weite Verbreitung
- besitzt **kommandobasierte** und graphische Benutzeroberfläche
- *In diesem Kurs:* „Abheben“ der graphischen Oberfläche, um näher an das System zu gelangen (es zu verstehen)

Aufbau und Aufgabenverteilung



Das UNIX-Prozesskonzept

- Beim Start eines Programms erzeugt der Kern einen Prozess.
- Wird das gleiche Programm mehrfach gestartet, so entsteht jeweils ein eigener Prozess.
- Das Prozesssystem ist **hierarchisch**, d.h. jeder Prozess hat einen **Elternprozess**, der ihn initiiert.

  Ausnahme: **Initializer**, den das System beim Booten erzeugt
- Verwaltung der Prozessattribute in einer **Prozesstabelle:**
  - für jeden Prozess eine Zeile
  - für jedes Attribut eine Spalte
  - kann mit dem Systemkommando `ps` abgefragt werden

Prozessattribute (Beispiele)

- Adressräume des Programms im Primär- und Sekundärspeicher
- Inhalte der Prozessor-Register (insbes. Befehlszähler)
- Prozesszustand (running/ready/sleeping/...)
- bislang verbrauchte Prozessorzeit (TIME)
- Prozesskennung (PID)
- Elternprozesskennung (PPID)
- Programmname (CMD)
- Benutzerkennung (UID)
- Terminal (TTY)

Beispiel

```
~$ ps
   PID  TTY          TIME  CMD
  3735  pts/0   00:00:00  bash
  4026  pts/0   00:00:00  ps
```

```
~$ ps -f
UID         PID   PPID   C  STIME  TTY          TIME  CMD
henning    3735   3727   0  21:26  pts/0   00:00:00  bash
henning    4027   3735   0  21:27  pts/0   00:00:00  ps -f
```

Die UNIX-Kommandozeile

Kommando [*Optionen*] [*Argumente*]

Kommando	eingebautes Shell-Kommando oder ausführbares Programm
Optionen	verändern die Funktionalität
Argumente	werden dem Kommando als Eingabedaten übergeben

```
ps          ps -f       ps -e -f          ps -ef
cp eins.pdf zwei.pdf    cp -r ordner1 ordner2
```