Betriebssysteme Vorlesung 1

Vorlesung: Dr. Peter Kemper
(email: peter.kemper@udo.edu)
Übungen+Praktika: Dirk Knabe
(email: knabe@fh-dortmund.de)

Heute:

Übersicht der Vorlesungsinhalte

Übersicht zu Rechensystemen & Betriebssystemen

Literatur: Stallings Kapitel 1,2 (Einführungskapitel)

Organisatorisches

- Sprechstunde: nach der Vorlesung + nach Vereinbarung
- Prüfung: Klausurzeitraum 1.7.-15.7.
- Unterlagen
  - Folienkopien http://ls4-www.informatik.uni-dortmund.de/QM/MA/pk/pie.html unter Teaching

    Dort sind Zipfiles mit PDF Dateien je Vorlesung abgelegt.
  - kein Skript -> Lehrbuch von Stallings,

    Literaturstelle auf der nächsten Folie

    Einige Exemplare sollten inzwischen vom IT-Center vorgehalten werden.

    Empfehlung http://www.WilliamStallings.com

Organisatorisches, Termine

- Terminplanung (auch auf der Webseite vorhanden)
- insgesamt 8 Veranstaltungen a 3h
- Termine:

  jeweils 14.00-16.15 Uhr
  - Vorlesung am 13.4., 27.4., 4.5., 18.5., 1.6., 8.6., 22.6., 29.6
  - Feiertage: 20.4. Osterwoche, 25.5., 15.6.
  - 30.6. Übungsklausur
- Übungen+Praktika: Herr Knabe

Literatur

W. Stallings; Operating Systems Internals and Design Principles, 4th ed, Prentice Hall 2001.

inzwischen 4. Ausgabe auch in deutscher Übersetzung!

inzwischen 5. Ausgabe in englischer Sprache

An typischen Lehrbüchern für Betriebssysteme herrscht kein Mangel:

- A. Silberschatz, P.B. Galvin, G. Garne; Applied Operating Systems Concepts, 1st ed, Wiley 2000.
- A. Silberschatz, P.B. Galvin; Operating Systems Concepts, 6th ed, Wiley 2003.
- A.S.Tanenbaum, Moderne Betriebssysteme, 2nd ed, Prentice Hall, 2002. dt Übersetzung von „Modern Operating Systems“.

Lernziele der Veranstaltung

- Sie werden wesentliche Konzepte kennen lernen, die in bestehenden, erfolgreich eingesetzten Betriebsystemen angewendet werden
  (Windows 98/NT/2000/XP, Unix, Linux).

Hinweise zur Arbeitstechnik:

- Es wird der Inhalt eines klassischen Lehrbuchs vermittelt, daher der dringende Hinweis: LESEN BILDET ... eine Alternative, Ergänzung, Vertiefung zur Vorlesung. Ziel der Vorlesung ist auch eine Hilfe zur Selbsthilfe.
- Bereiten Sie die Vorlesung möglichst vor / nach!
  Stallings bietet im Anschluss an Kapitel 3,4,... jeweils einen zusammenfassenden Abschnitt mit zentralen Begriffen und Kontrollfragen zum Verständnis.
- Beteiligen Sie sich aktiv an den Übungen!
- Formulieren Sie Unklarheiten als konkrete Fragen, stellen Sie Fragen in der Vorlesung und in den Übungen!

Danksagung - Credits

- Folien nach Vorlagen von
  - Stallings
  - Roy
  - Marchand
  - Liedtke
  - Knottenbelt, Magee, Kramer
  - Kelly
  - Burton
  - ...

Übersicht

- Ein Betriebssystem - Was ist das ?
- Aufgaben und Anforderungen
  - Prozeß Management
  - Speicher Management
  - Prozessor Scheduling
  - E/A Management, Festplatten Scheduling
  - Datei Management
  - Netzwerke
  - Sicherheit

Nutzen

- Nach dieser Veranstaltung wissen Sie,
  - was ein Betriebssystem ist und
  - welche Aufgaben ein Betriebssystem erfüllen muss.

Einführungsbeispiel

- Wir betrachten die aktuell laufende Vortragspräsentation mittels Laptop und Beamer.

  Welche Hardware & Softwarekomponenten sind an der Entstehung der Präsentation beteiligt ?
- Hardware
  - Beamer als Ausgabemedium
  - Monitor als Ausgabemedium
  - Tastatur als Eingabemedium
  - Festplatte als Datenträger für die Präsentationssoftware und die Datei mit den Präsentationsinhalten
  - Hauptspeicher als Zwischenspeicher für die aktuell laufende Präsentation als Prozeß
  - Bus als Übermittler von Daten zwischen Komponenten
  - Prozessor als aktive Einheit, die den Prozeß der Präsentation abarbeitet
- Software
  - Präsentationssoftware

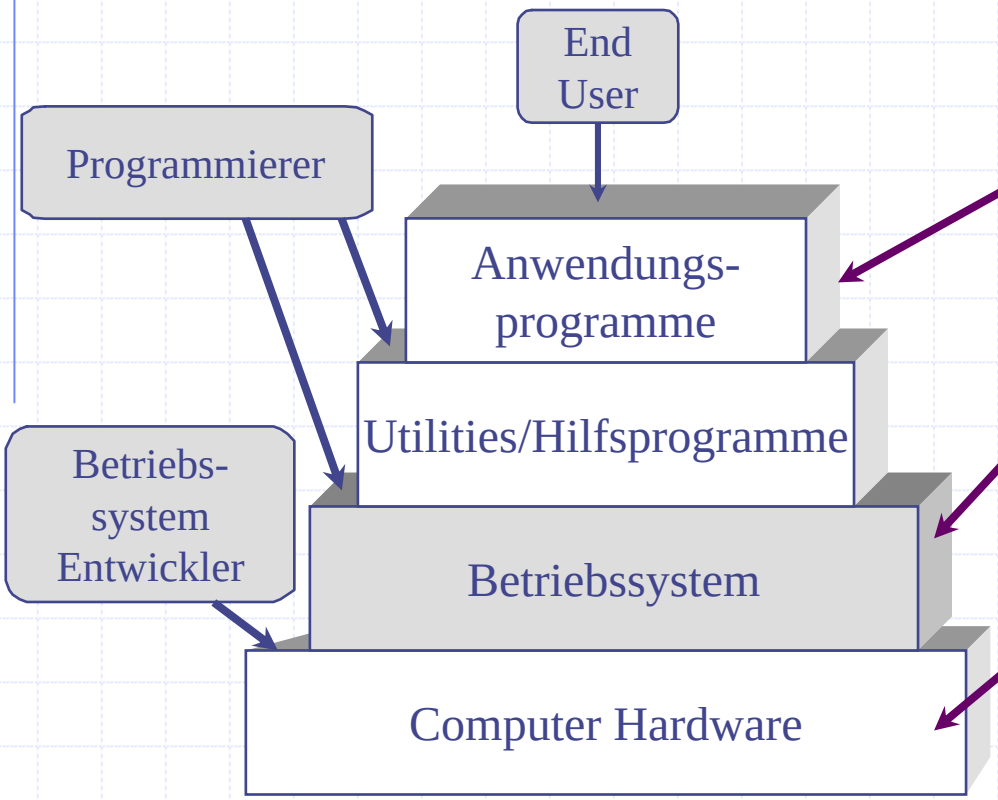
Was ist ein Betriebssystem ?

- Ein Betriebssystem (OS: Operating System) ist Software, ein Programm. Es erlaubt andere Programme auf einer Hardwareplattform durchführen zu können.
- Ein Betriebssystem verwandelt Hardware in eine virtuelle Maschine. Diese virtuelle Maschine bietet alle Funktionen zum Betrieb von Anwendungsprogrammen. Die Besonderheiten zur Bedienung der jeweiligen realen Hardwareplattform werden durch das Betriebssystem verdeckt.
- Ist das nötig ?

  Durch ein OS werden Anwendungsprogramme zu einem eigenständigen Produkt und handelbar

  - eine kommerziell interessante Angelegenheit!

Schichten (auch Sichten) eines Computer Systems

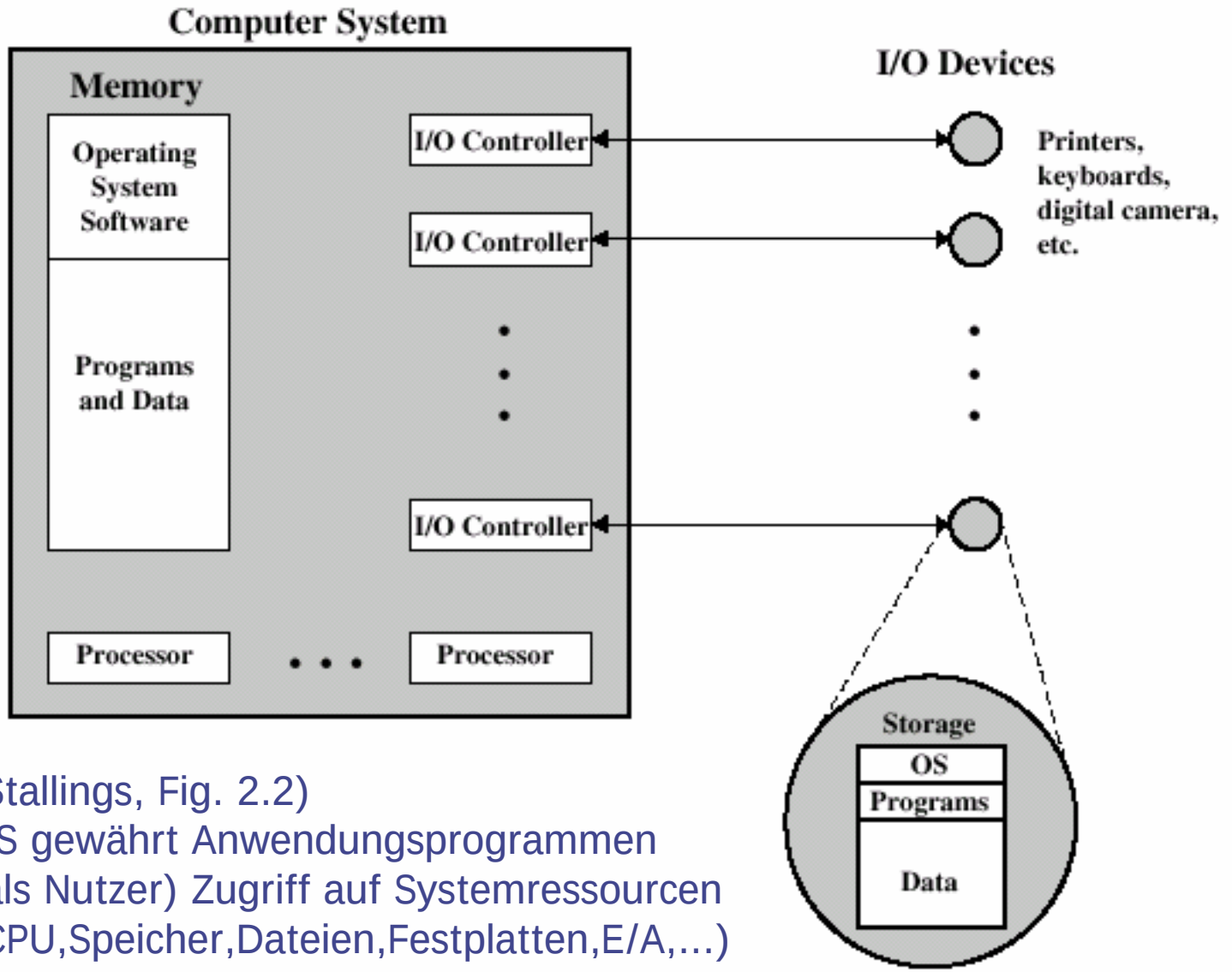


Beispiel:
Powerpointpräsentation
Powerpoint nutzt
- Dateisystem,
- logische E/A Geräte

Powerpointprozeß wird vom OS verwaltet, Dateisystem, E/A Geräte werden unter Nutzung realer Hardware verfügbar gemacht.

Das Betriebssystem vermittelt also zwischen Anwendungsprogrammen und realer Hardware. Z.B. Powerpoint hat eine „Save“ Funktion zum Speichern einer Präsentation in einer Datei. Das Betriebssystem stellt hierzu elementare Operationen zum Lesen / Schreiben von Dateien zur Verfügung und verwaltet Dateien.

Das Betriebssystem als Ressourcenverwalter



(Stallings, Fig. 2.2)

OS gewährt Anwendungsprogrammen
(als Nutzer) Zugriff auf Systemressourcen
(CPU,Speicher,Dateien,Festplatten,E/A,…)
und sichert ordnungsgemäßen Ablauf auch bei mehreren Prozessen.

Anforderungen an ein Betriebssystem

- Prozeßmanagement (Prozeß = in Bearbeitung befindliches Programm + Daten)

  Multiprogramming: Ein modernes OS erlaubt die gleichzeitige Bearbeitung von mehreren Prozessen. Dazu müssen Prozesse verwaltet werden, z.B. OS muss Prozessbearbeitung nachhalten, so dass die Bearbeitung unterbrochen und zu einem späteren Zeitpunkt fortgesetzt werden kann.
- Speicher Management

  Prozesse müssen ausreichend Speicher erhalten, dürfen sich nicht stören, müssen aber auch Speicher teilen können.
- Prozessor Scheduling

  Der Prozessor muss als zentrale Ressource zwischen den Prozessen sinnvoll (gerecht, fair ?) aufgeteilt werden.
- E/A Management, Festplattenscheduling

  Ein/Ausgabegeräte müssen so verwaltet werden, dass Prozesse sie (möglichst ungestört voneinander) nutzen können.
- Datei Management

  Auf dem rohen Festplattenspeicher muss ein Dateisystem betrieben werden.
- Netzwerke

  Prozesse müssen über Netzwerke hinweg kommunizieren können.
- Sicherheit: (Vertraulichkeit, Integrität, Verfügbarkeit, Authentizität)

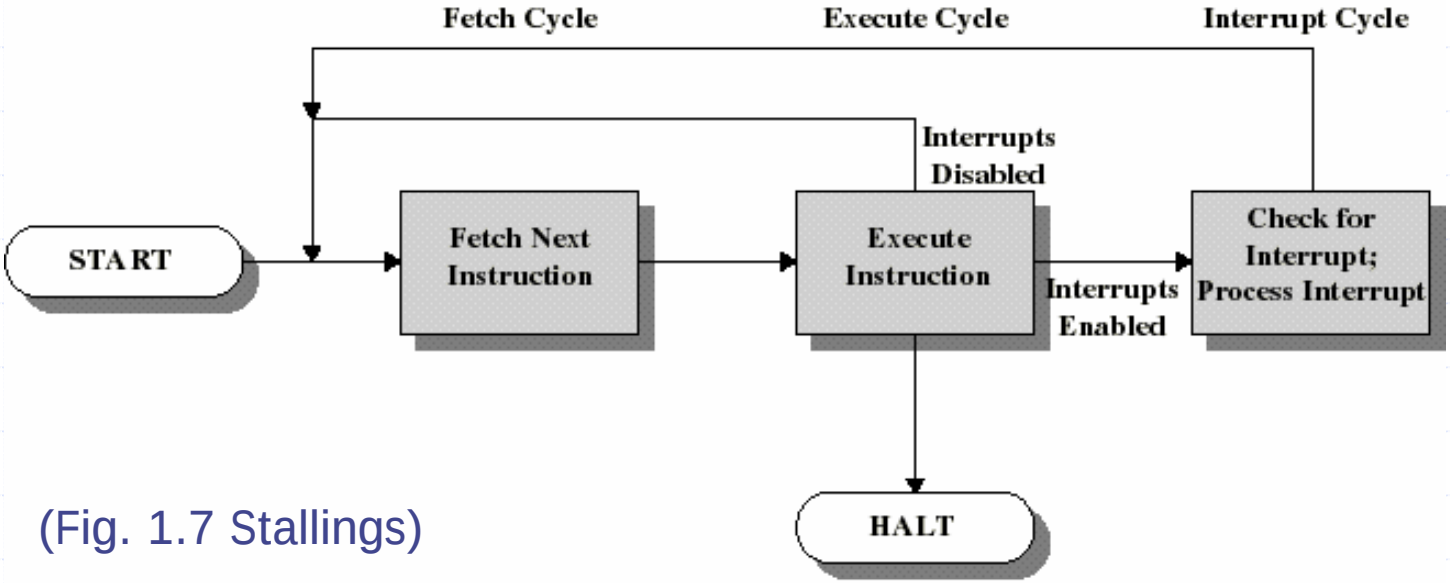
Prozeßmanagement

- Prozesse (Prozeß = in Bearbeitung befindliches Programm + Daten) werden durch ein Prozeßimage im Speicher repräsentiert:
  - Verwaltungsinformation: Angaben zum aktuellen Status der Bearbeitung des Prozesses, der belegten Ressourcen, Informationen zum Fortsetzen der Bearbeitung in der CPU
  - Code: Liste der Programmzeilen, die abgearbeitet werden müssen
  - Stack: Aufrufstack der in der aktuellen Situation aufgerufenen Funktionen mit ihren Parameterwerten und lokalen Variablen
  - Daten: Werte von Variablen und Objekten, die nicht funktionslokal existieren.
- Prozesse und das Betriebssystem selbst werden im Wechsel durch die CPU bearbeitet. Wie erfolgt die Steuerung für derlei Wechsel ?

Zur Erinnerung: Instruktionszyklus + Interrupts der CPU

CPU prüft nach jeder Instruktion: liegt Interruptsignal vor ?

- falls nein: nächste Instruktion laden und ausführen
- falls ja: Ausführung des aktuellen Programms unterbrechen und Interrupt Handler ausführen. Man unterscheidet
  - synchrone Ereignisse: Exception (aktueller Prozeß bewirkt Fehlersignal), Seitenfehler (Prozeß fordert noch nicht verfügbaren Teil des Speichers an), ...
  - asynchrone Ereignisse: Timer (regelmäßige Unterbrechung, damit OS Chance hat, andere Prozesse zu bearbeiten), E/A Signale bzgl des Abschlusses von delegierten E/A Aufgaben



(Fig. 1.7 Stallings)

Prozeßmanagement -> CPU Scheduling

Multiprogramming erfordert, dass OS die CPU (CPUs) zuteilt.

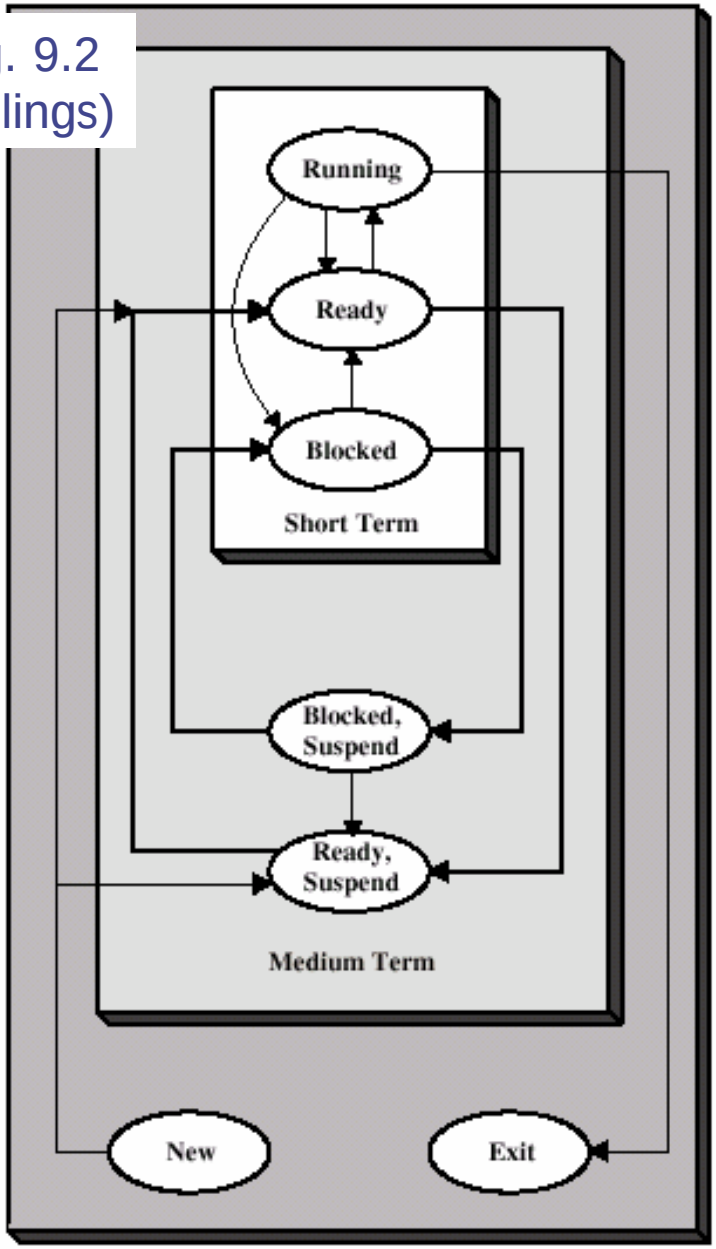
Randbedingungen:

- Fairness, d.h. vorhandene Ressourcen sollten unter den Jobs gleicher Wichtigkeit gleichmäßig (fair) und sinnvoll aufgeteilt werden.
- Differenziert bzgl „Responsiveness“, je nach Art eines Jobs und seiner gehaltenenen Ressourcen sollten Prioritäten gesetzt werden. Z.B. interaktive Prozesse und Prozesse, die auf Eingaben warten sollten nicht länger als unbedingt nötig blockiert werden.
- Effizienz, d.h. das OS sollte den Durchsatz maximieren, Antwortzeiten minimieren und möglichst wenige Anwender abweisen. (ACHTUNG: Ziele sind konträr)
- Echtzeitanforderungen (Realtime), Prozesse mit harten Terminvorgaben sollten möglichst fristgerecht beendet werden.

- Man unterscheidet 3 Arten von Scheduling nach dem Zeithorizont:
  - langfristig: wieviele & welche Aufgaben werden mit Prozessen bearbeitet
  - mittelfristig: welche Prozesse werden ein/ausgelagert (Swapping)
  - kurzfristig: (Dispatching) welcher Prozeß soll bei einem Wechsel bearbeitet werden, reagiert auf Interrupts (Clock, I/O), Systemcalls, Signale

CPU Scheduling

(Fig. 9.2 Stallings)



Ein Betriebssystem unterscheidet Prozesse nach ihrem Zustand:

Running = wird aktuell durch die CPU bearbeitet

Ready = bereit zur Bearbeitung durch die CPU, wartet

Blocked = blockiert, wartet auf externes Ereignis (typischerweise I/O)

Suspend = Prozessimage ist auf die Festplatte ausgelagert und muß vor Bearbeitung erst in den Hauptspeicher geladen werden.

New = neuer Prozeß

Exit = abgearbeiteter Prozeß

Scheduler arbeiten typischerweise mit einem Zeitscheibenverfahren (Round Robin) und Prioritäten, die zur Laufzeit variiert werden.

Prozeßmanagement -> Speichermanagement

Multiprogramming erfordert, dass der laufende und weitere, bereite Prozesse genügend(?) Speicher erhalten.

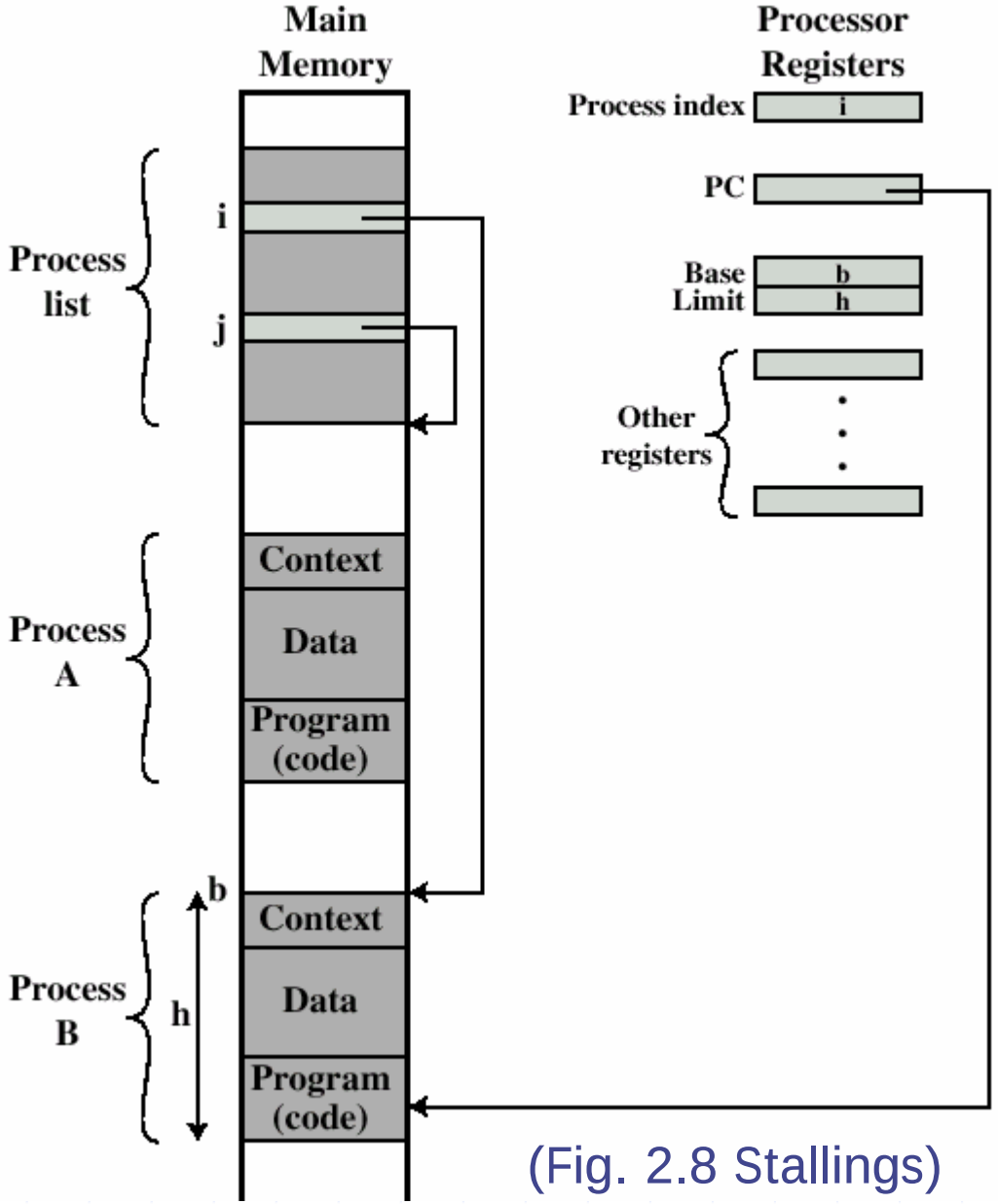
Randbedingungen:

- Isolierung von Prozessen, d.h. unabhängige Prozesse dürfen untereinander keinen Zugriff auf Daten oder Speicher haben
- Automatische Allokation und Verwaltung von Speicher über die Speicherhierarchie hinweg, d.h. ein Programm erzeugt Objekte, fordert Speicherplatz zur Laufzeit an, ohne dass Restriktionen innerhalb der realen Speicherhierarchie explizit berücksichtigt werden müssen.
- Schutz und Zugriffskontrolle auf geteilte Speicherbereiche müssen vom OS gleichermaßen bewirkt werden.

Wichtigstes Konzept des Speichermanagement:

Virtual Memory: logische und physikalische Adressen werden getrennt, Speicher wird in kleinen Teilen (Seiten/Segmentweise) verwaltet.

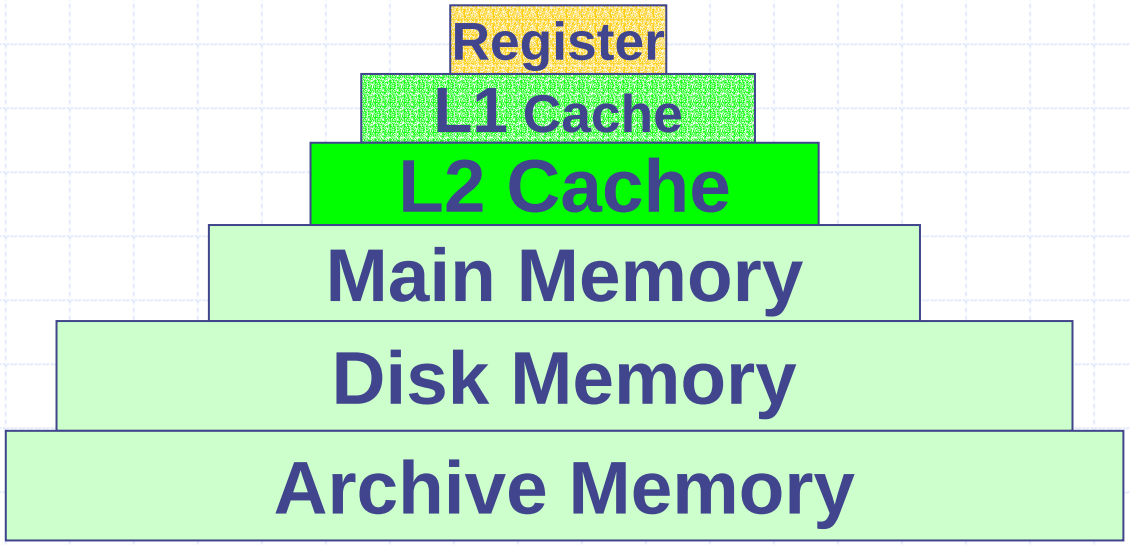
Prozeßmanagement -> Speichermanagement



(Fig. 2.8 Stallings)

- Hauptspeicher wird als ein logischer Block mit Adressen 0,1,...,n für einzelne Speicherzellen betrachtet.
- Einzelne Prozesse erhalten Teile des Speichers für ihr Prozessimage.
- OS stellt benutzte Teile des Prozessimages jeweils zur Verfügung. Beim Zugriff werden logische Adressen auf physikalische abgebildet.
- Virtueller Speicher erlaubt, dass ein Prozess oder mehrere Prozesse mehr Hauptspeicher nutzen können, als real vorhanden ist.

Speicherhierarchie



preiswerter,
langsamer,
größer,
weniger häufig
genutzt

Zugriffszeiten unterscheiden sich von einer Stufe zur nächsten um Größenordnungen !!!

Für die Performance ist eine geschickte Nutzung von schnellen, aber leider kleinen Speichermedien wichtig.

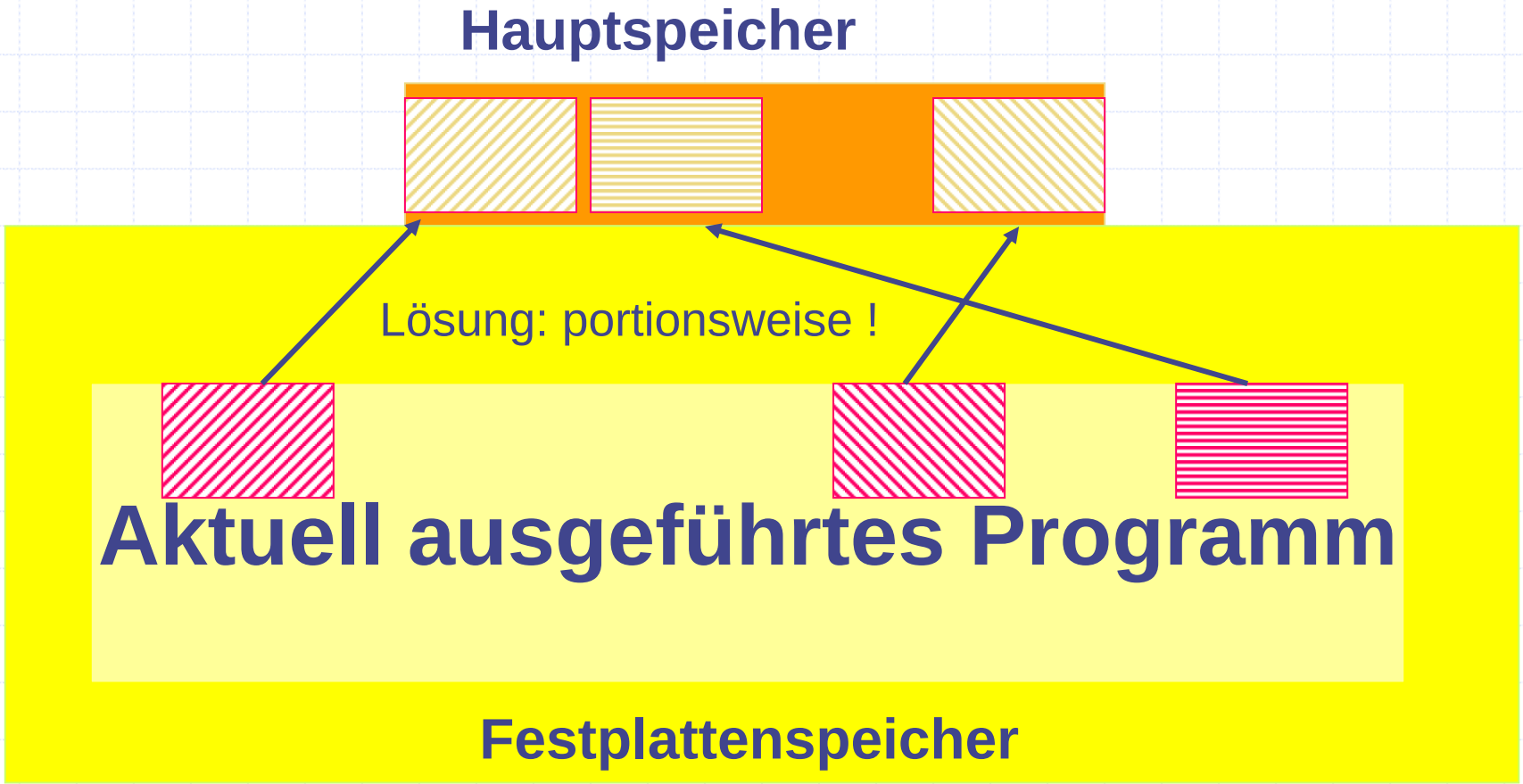
Grundidee: häufig benutzte Daten/Teile eines Prozeßimages sind hoch angesiedelt, selten benutzte Daten werden auf der Festplatte gehalten.

=> bewirkt eine Anzahl von Entscheidungen, für die ein OS Regeln haben muss, z.B. welche Seiten sollen beim Einlagern neuer Seiten ersetzt werden ?

Grundidee zum virtuellen Speicherkonzept

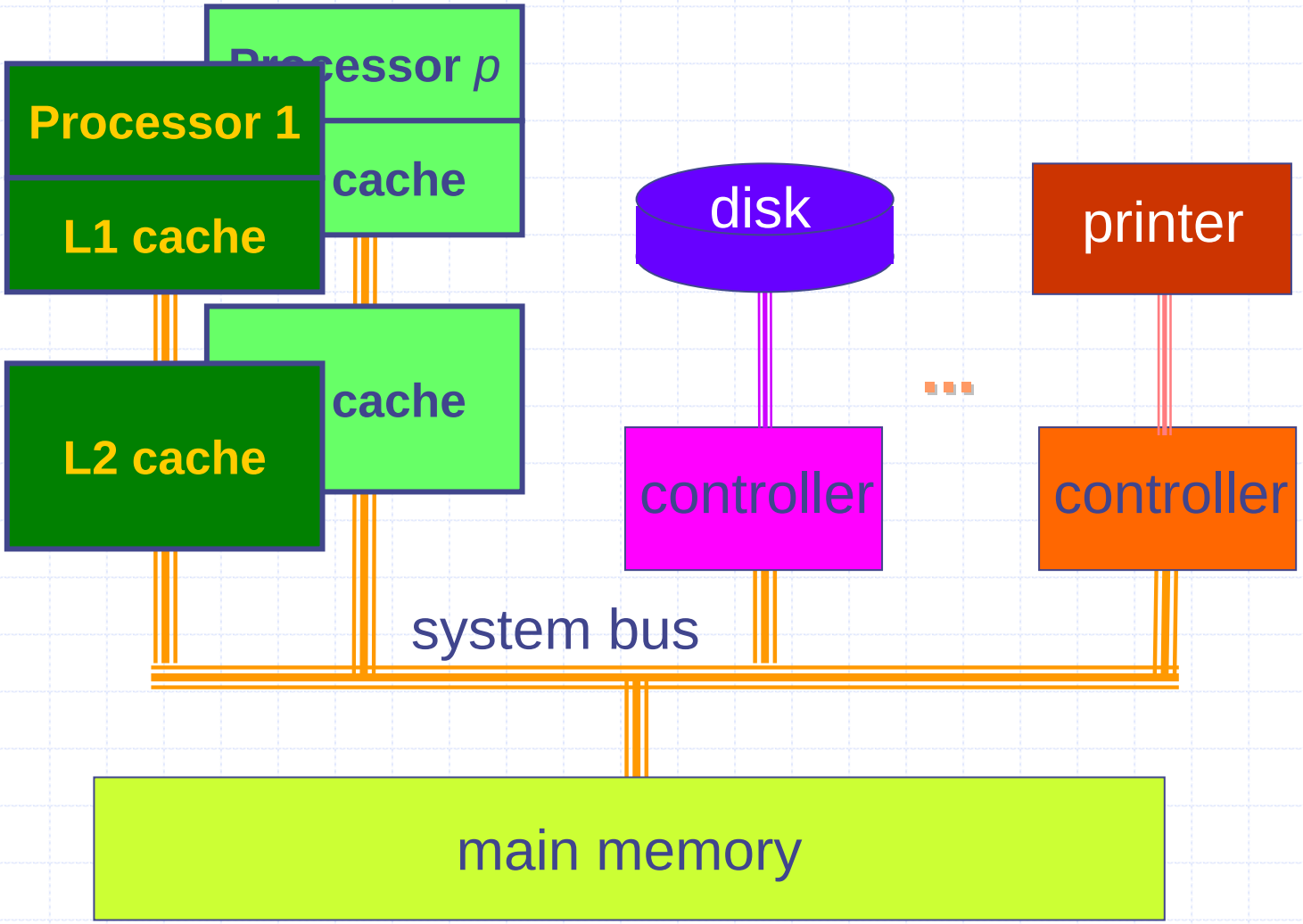
z.B. Paging, Speicherseiten sind alle gleich groß,

Grundidee zum virtuellen Speicherkonzept



Gleiche Idee weiter gedacht: wenn nur Teile eines Prozesses im Hauptspeicher sein müssen, können insgesamt natürlich viele Prozesse in Teilen vorliegen und dennoch bearbeitet werden. Fehlende Teile werden bei Bedarf von der Festplatte beschafft (mit Zeitverzögerung).

Multi Prozessor System



Prozeßmanagement -> E/A Management

Das Betriebssystem verdeckt weitgehend die Besonderheiten der jeweiligen Hardwareeinheiten eines Rechners, insbesondere die Vielzahl Ein-/Ausgabegeräte (E/A Geräte, I/O Devices) werden hinter weitgehend homogenen Programmierschnittstellen verborgen.

Für die Realisierung benötigt ein OS für jeden Gerätetyp eine passende Steuerungssoftware (Gerätetreiber, Device driver).

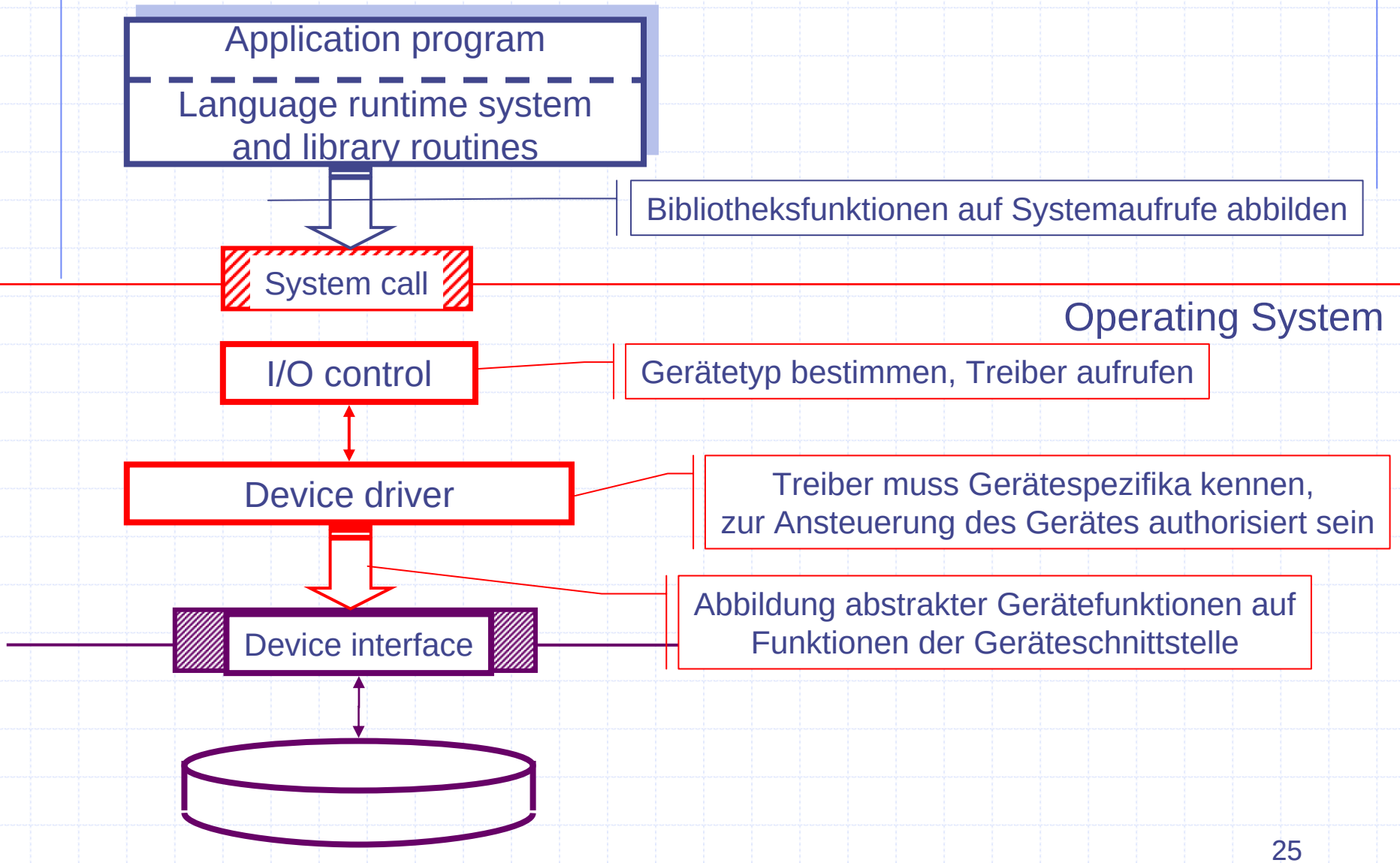
Aufgrund der großen Zeitunterschiede zwischen der Bearbeitung in der CPU und in E/A Geräten erfolgt E/A typischerweise asynchron (z.B. Direct memory access DMA), wodurch zusätzliche Kommunikationsmöglichkeiten erforderlich werden (Interrupts).

Wesentliche Ziele im OS Entwurf sind:

Effizienz & Allgemeingültigkeit, d.h. E/A muss möglichst schnell sein und eine einheitliche Sichtweise muß die Programmierung von E/A Funktionen und die Implementierung E/A Software kennzeichnen.

Typische Konzepte sind: Puffern, Caching

Typische Schichten eines I/O-Subsystems



Dateimanagement

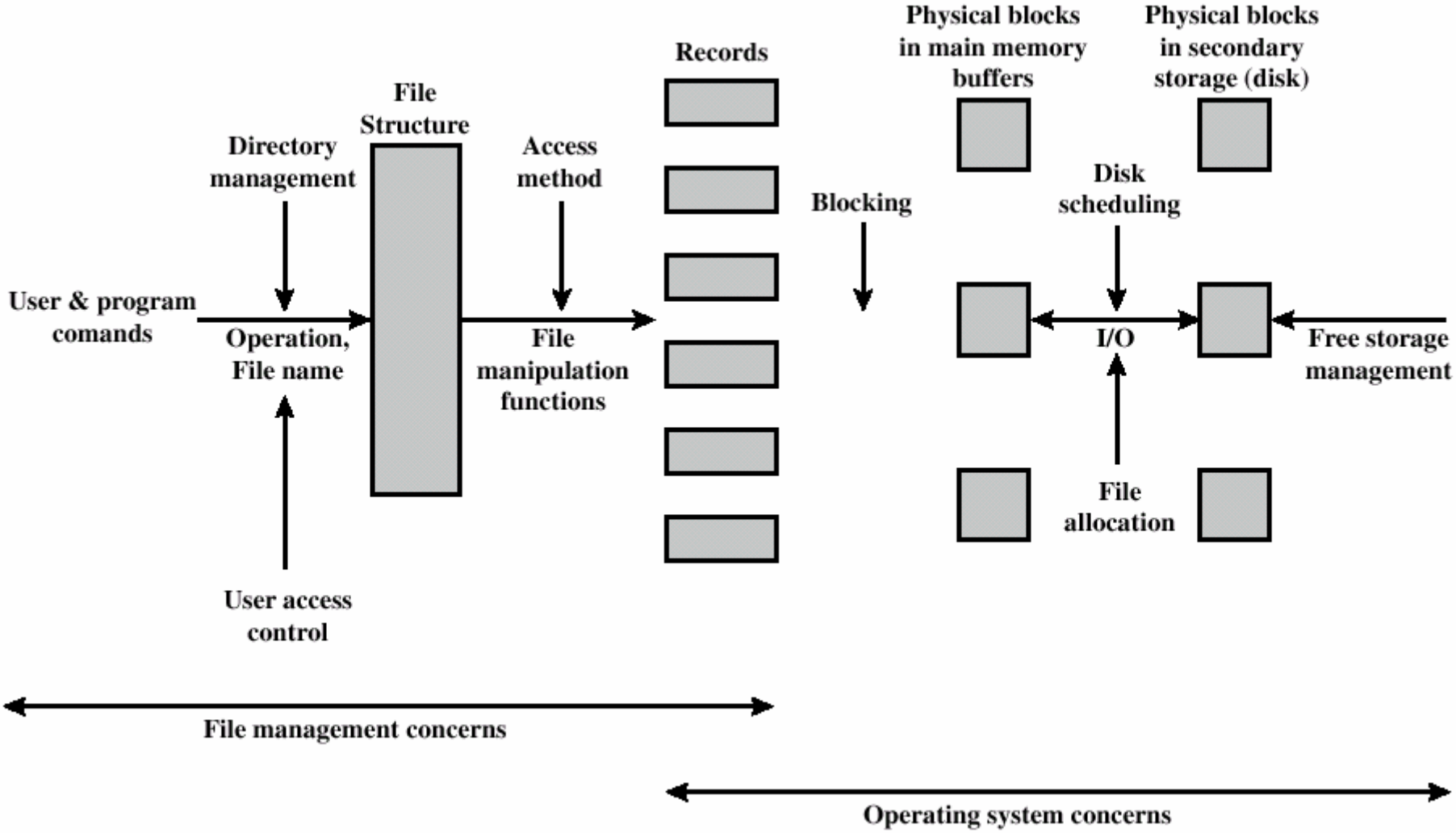
Dateien bieten die Möglichkeit, die Resultate eines Programms über das Ende des Prozesses hinaus dauerhaft zu speichern.

Das Dateimanagement hat die Aufgabe, das logische Konzept der Datei, das durch ein OS angeboten wird, zu realisieren.

Typische Verwaltungsstrukturen sind baumartige Directories, deren Wege von der Wurzel durch verschiedene Unterdirectories bis zur konkreten Datei einen eindeutigen Dateinamen ergeben, zu dem das OS dann die zugehörigen Inhalte speichern muß.

Anforderungen sind: Effizienz (bzgl Zeit und Platz), Zuverlässigkeit.

Dateimanagement, einige Aspekte



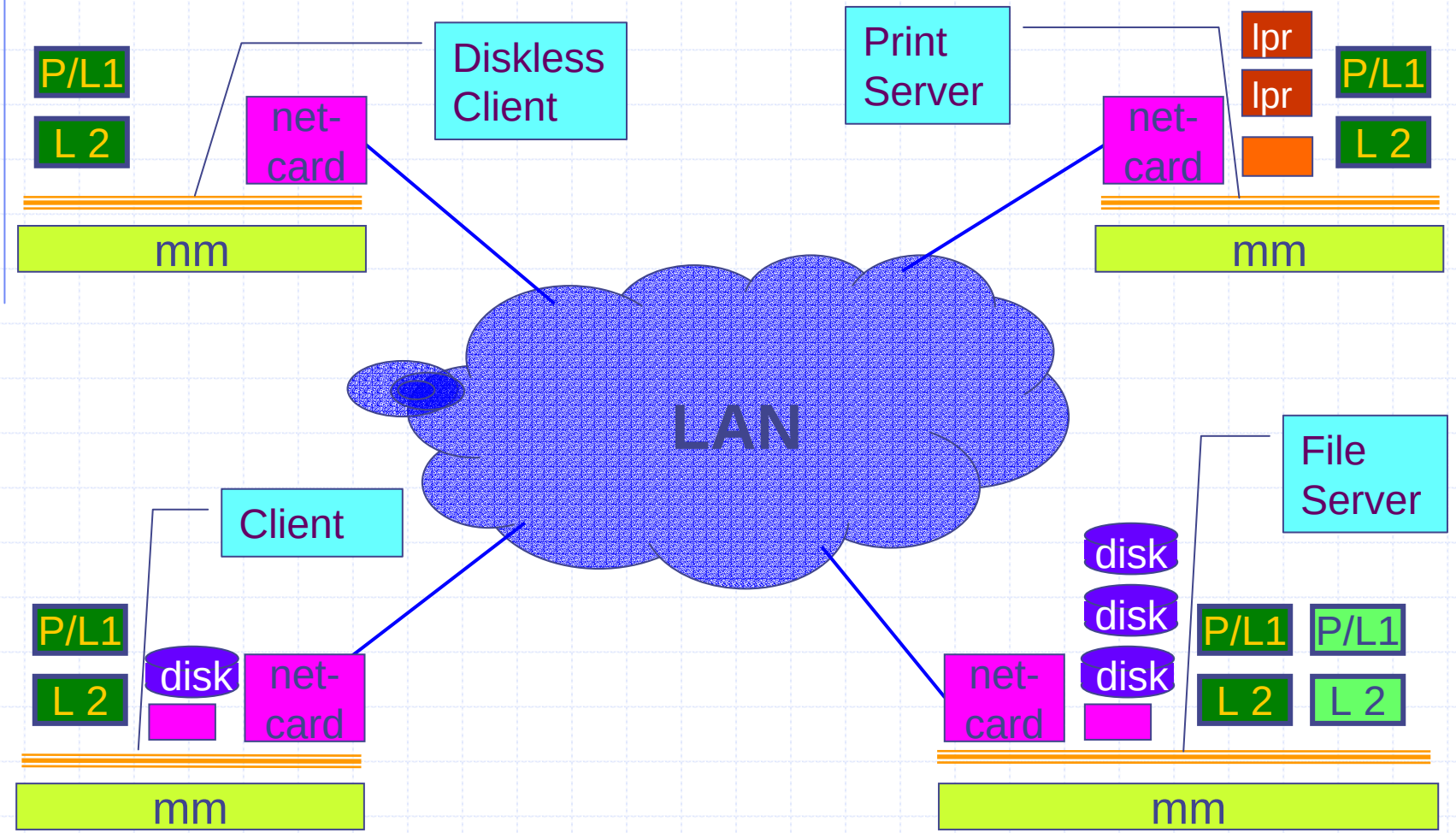
Verteilte Systeme, Netzwerke

Unterstützung von Client/Server Architekturen durch ein Betriebssystem:

- Message passing, zwischen Prozessen werden Nachrichten übermittelt
- Remote Procedure Calls, Prozesse greifen auf Funktionen zu, die nicht vom lokalen Rechner durchgeführt werden, sondern an einen externen Server weitergeleitet werden und dort ausgeführt werden. Die Ergebnisse dieser Berechnungen müssen natürlich übermittelt werden.

Verteilte Betriebssysteme gehen noch einen Schritt weiter, sie werden auf einem Netzwerk realisiert und wirken wie eine lokale „virtuelle“ Maschine. Sie erlauben die Migration von Prozessen.

Lokales Client-Server Netzwerk



Sicherheit

- Anforderungen
  - Vertraulichkeit von Daten: Daten dürfen nur berechtigten Teilnehmern zur Kenntnis gelangen, d.h. lesen / ausdrucken, aber auch bereits die Existenz der Daten
  - Integrität von Daten: Erzeugung, Änderung von Inhalten und Status und Zerstörung darf nur durch Berechtigte erfolgen
  - Verfügbarkeit von Daten: berechtigte Zugriffe / Operationen müssen verfügbar / durchführbar sein
  - Authentizität von Teilnehmern: Rechensystem muss Identität des Teilnehmers, des Auslösers einer Aktion verifizieren können
- Anforderungen gelten in Teilen sinngemäß auch für Hardware (Verfügbarkeit), Software und Kommunikationsverbindungen.

Sicherheit

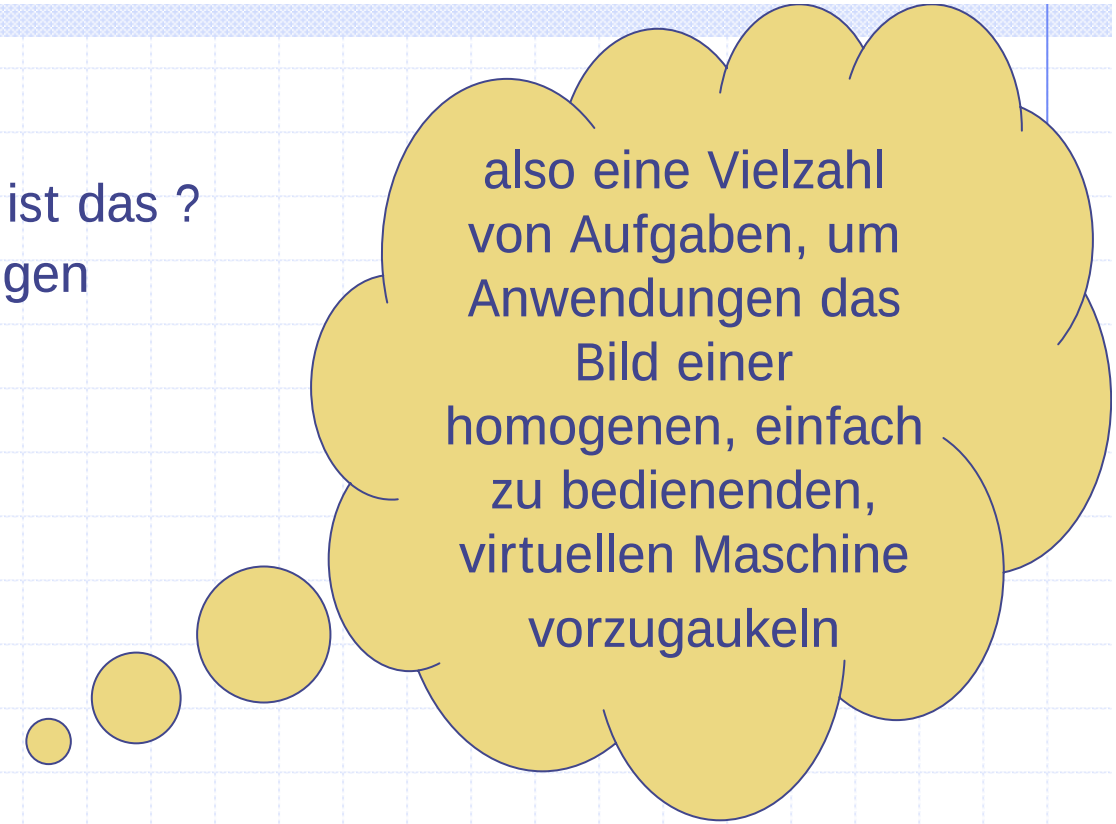
- Schutzmechanismen
  - Speicherschutz wg Multiprogrammings durch virtuelle und physikalische Adreßräume
  - Benutzerorientierte Zugriffskontrolle durch IDs und Passwörter
  - Datenorientierte Zugriffskontrolle durch Vergabe von Zugriffsrechten für Einzelne oder Gruppen je Objekt / Datei / ...
- Angreifer (Intruders)
  - unterscheidbar nach externen und internen Angreifern

  Gegenmaßnahmen:
  - Sicherung der Password Dateien durch Einweg-Verschlüsselung und Zugriffskontrolle
  - Intrusion Detection anhand statistischer Anomalien im Benutzerverhalten oder anhand von Regeln zur Beschreibung des Verhaltens von Angreifern
- Bösartige Software (Malicious Software, Malware)
- Vertrauenswürdige Systeme (Trusted Systems)
- Objektorientierte Sicherheitskonzepte, z.B. in Windows 2000

Zwischenstand

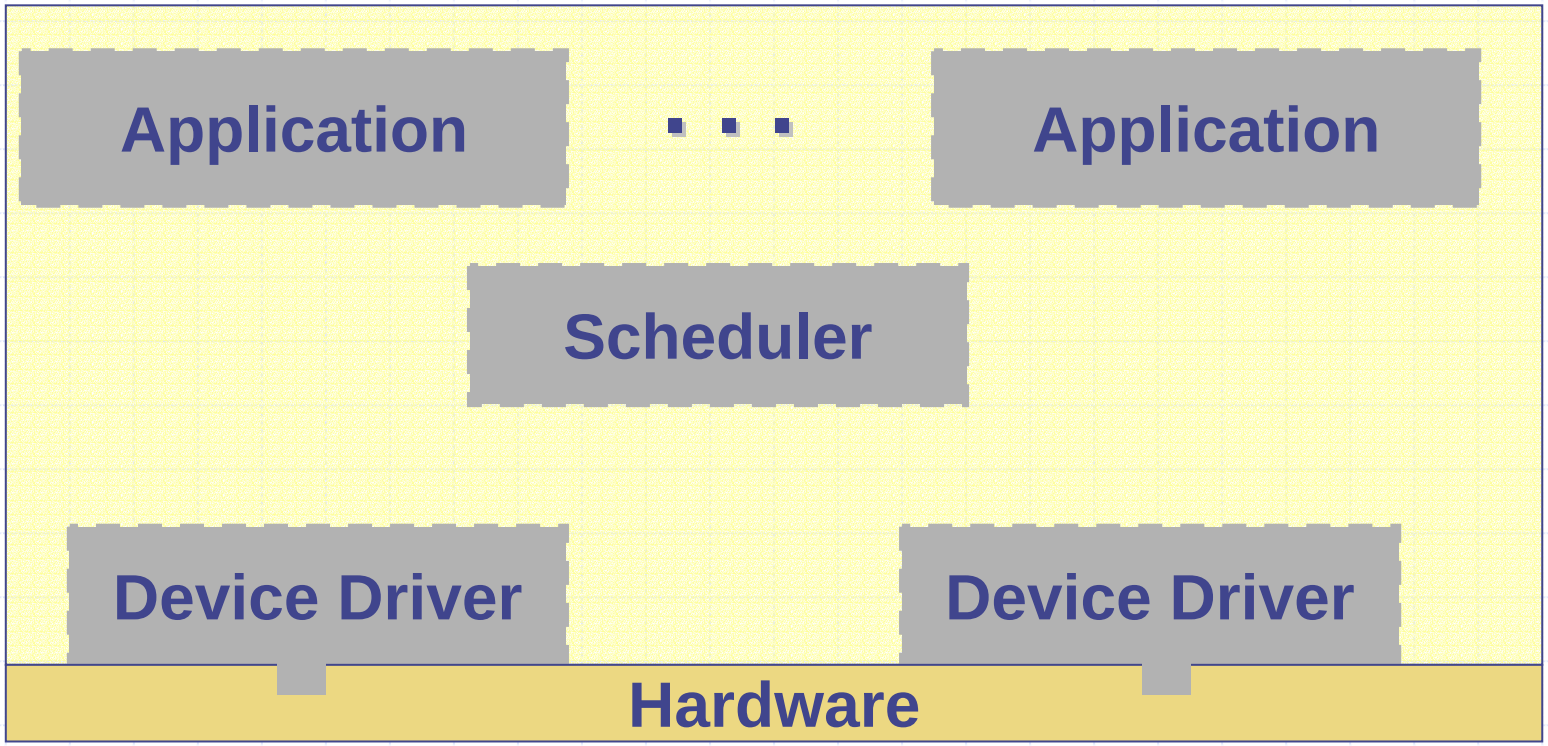
- Ein Betriebssystem - Was ist das ?
- Aufgaben und Anforderungen
  - Prozeßmanagement
  - Speicher Management
  - Prozessor Scheduling
  - E/A Management, Festplattenscheduling
  - Datei Management
  - Netzwerke
  - Sicherheit

  jeweils unter den Randbedingungen Effizienz, einfacher Benutzbarkeit, einfacher Wartbarkeit / Weiterentwicklung



- einige Beispielarchitekturen für Betriebssysteme
  - Monolithisches System, Monolithischer Kernel,
  - Schichtenmodell (Unix),
  - Mikrokernelarchitektur (Windows 2000)

OS Variante: Monolithisches System



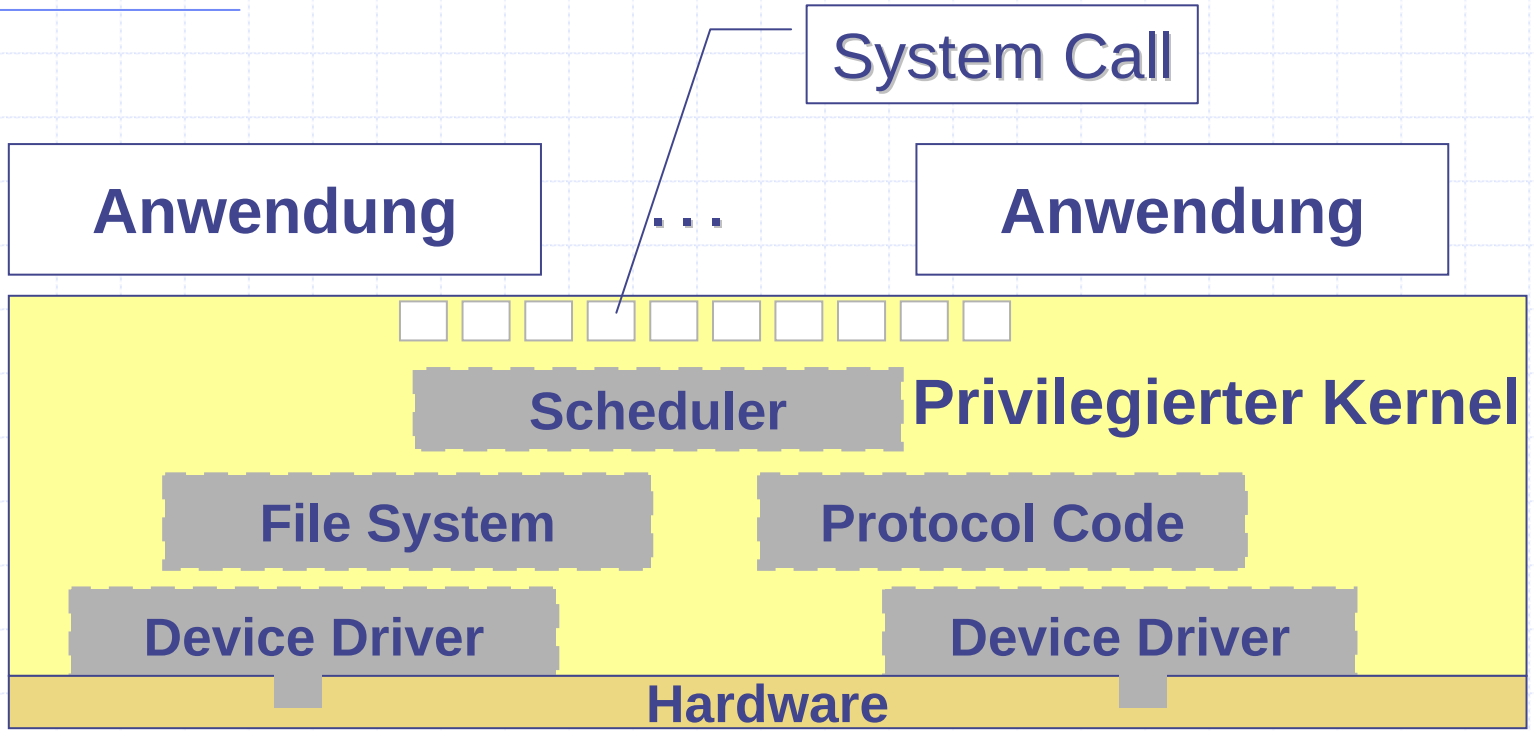
historisch, 1. Ansatz für sehr einfache kleine Systeme

Bewertung:

Schnell, kein Zugriffsschutz.

Jede Anwendung kann System allein kontrollieren und zu Fall bringen.

OS Variante: Monolithischer Kernel

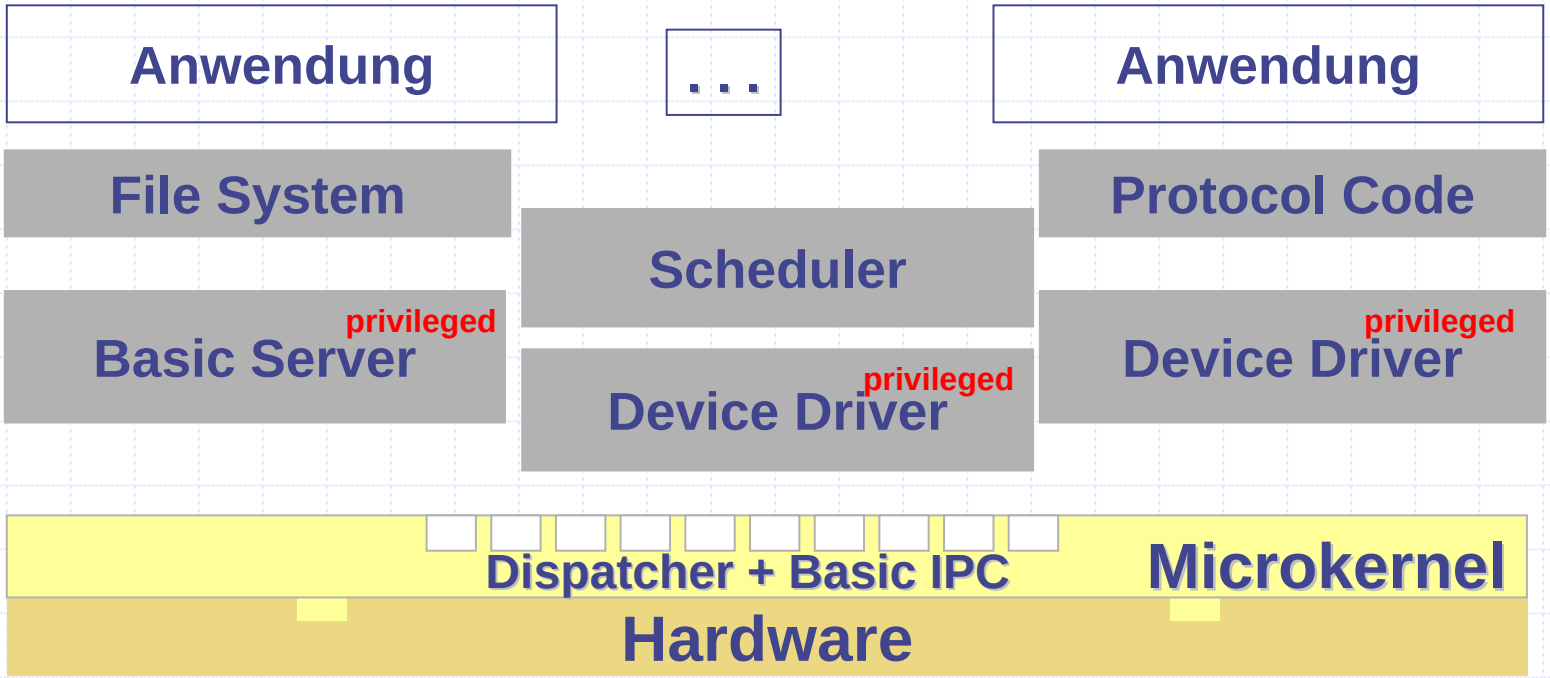


2. Ansatz, Unterscheidung von System- und Anwendungssoftware

Bewertung:

- Mehrzahl der OS Dienste im "privilegierten Kernel": =>
- Anwendungen: geschützt gegenüber anderen Anwendungen (virtueller Adreßraum), Anwendungen müssen sich auf Kernel verlassen
- Kernel Komponenten: geschützt gegen Anwendungen, aber nicht geschützt gegenüber anderen Kernel Komponenten.

OS Variante: Mikrokernelarchitektur



3. Ansatz: objektorientiertes Design, modulare Struktur, minimaler Kernel mit wenigen Operationen, Kommunikation über Nachrichtenaustausch

Bewertung:

- klare Abgrenzung von Teilaufgaben, dynamischer Austauch von Teilen
- natürliche Erweiterung/Integration in verteiltes System
- ACHTUNG: PERFORMANCE IST KRITISCH

Beispiel für modifizierte Mikrokernelarchitektur

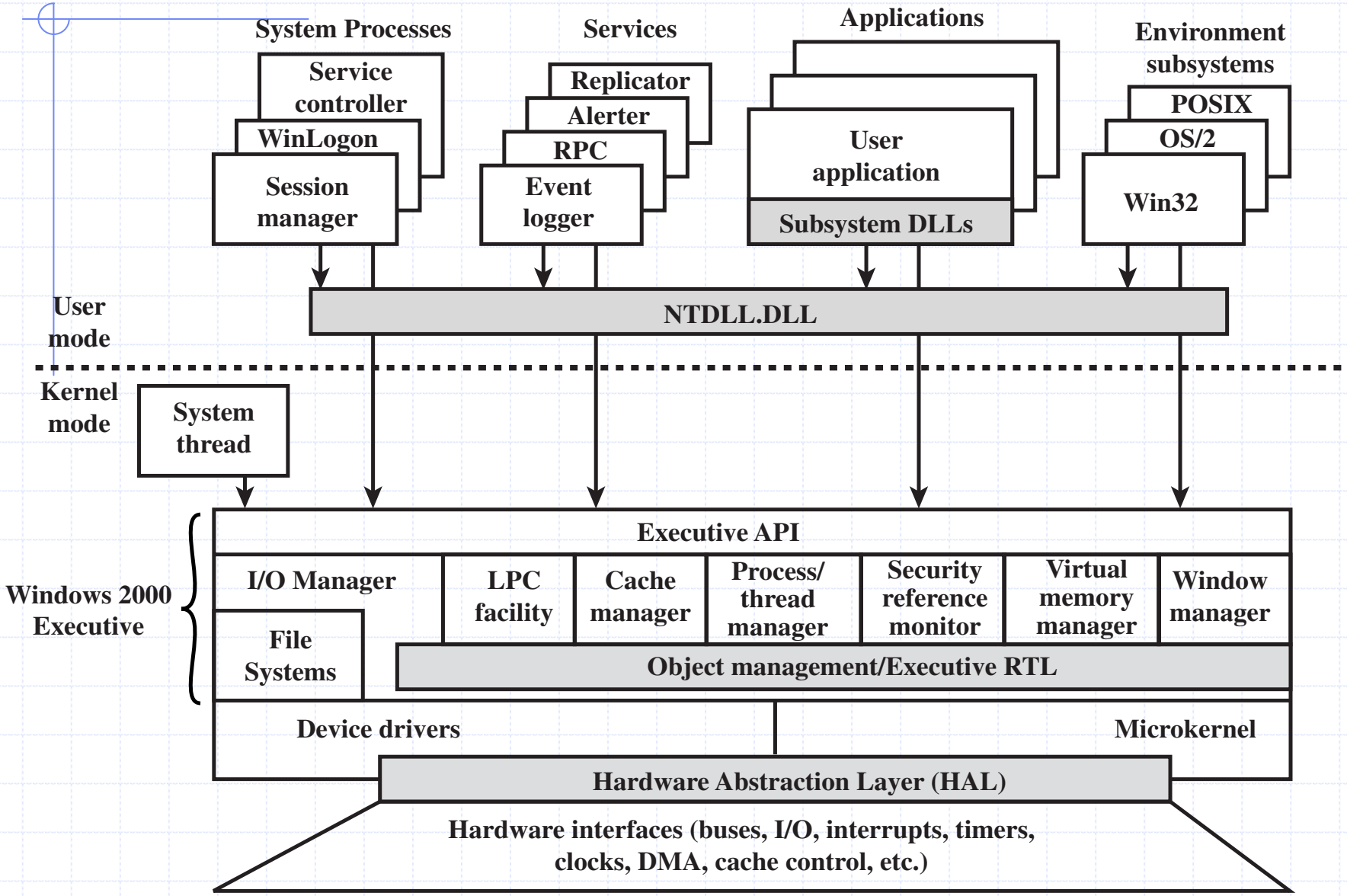
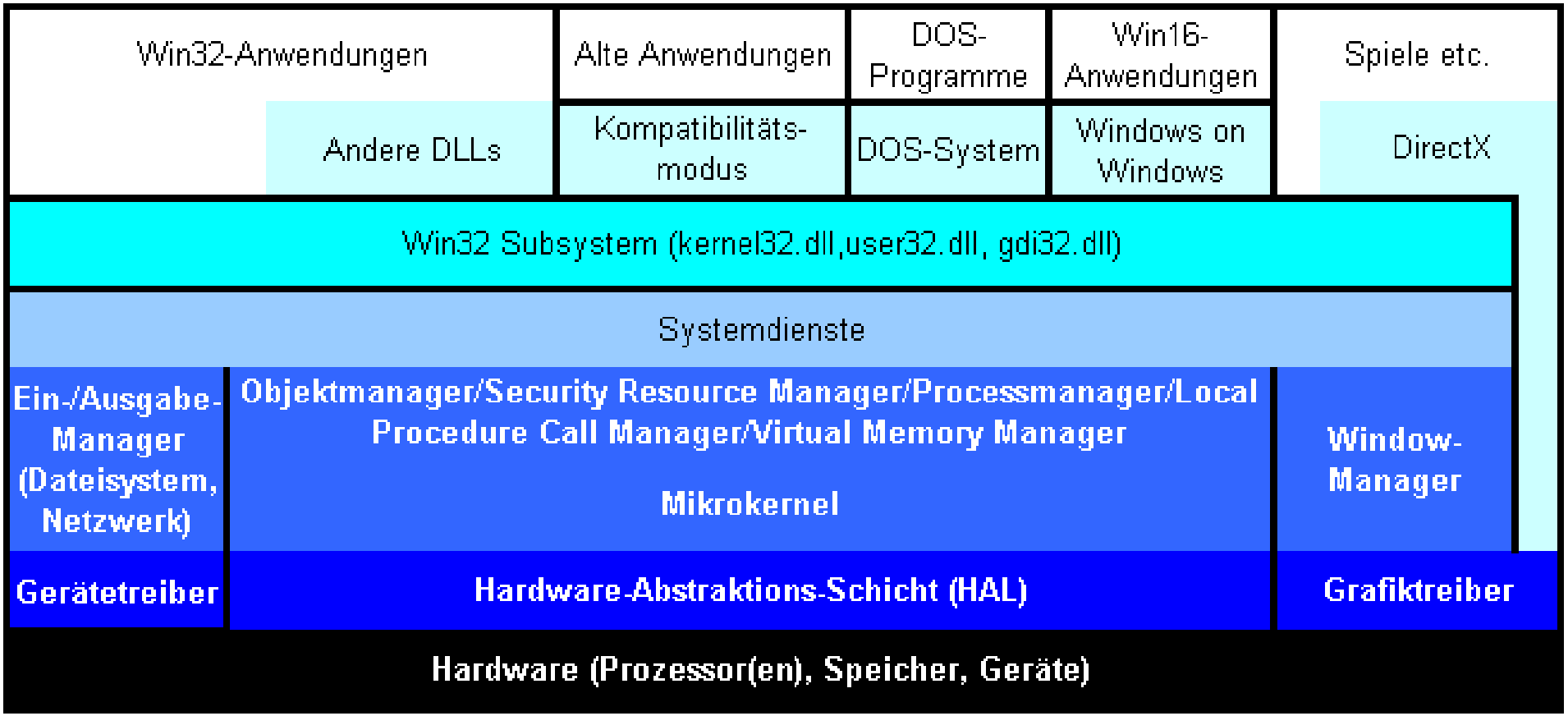


Figure 2.13 Windows 2000 Architecture

Etwas übersichtlicher dargestellt ...



Übersicht Windows XP (Windows NT 5.1)

Schichtenmodell eines OS

- Klassische Software Struktur

  jede Schicht nutzt Dienste niederer Schichten und bietet Dienste für höhere Schichten an
- jede Schicht realisiert einen (disjunkten) Teil der Gesamtfunktionalität
- Ziel ist Beherrschung von Komplexität durch Aufteilung und Kapselung
- Problem in der Realisierung: zirkuläre Nutzungsverhältnisse

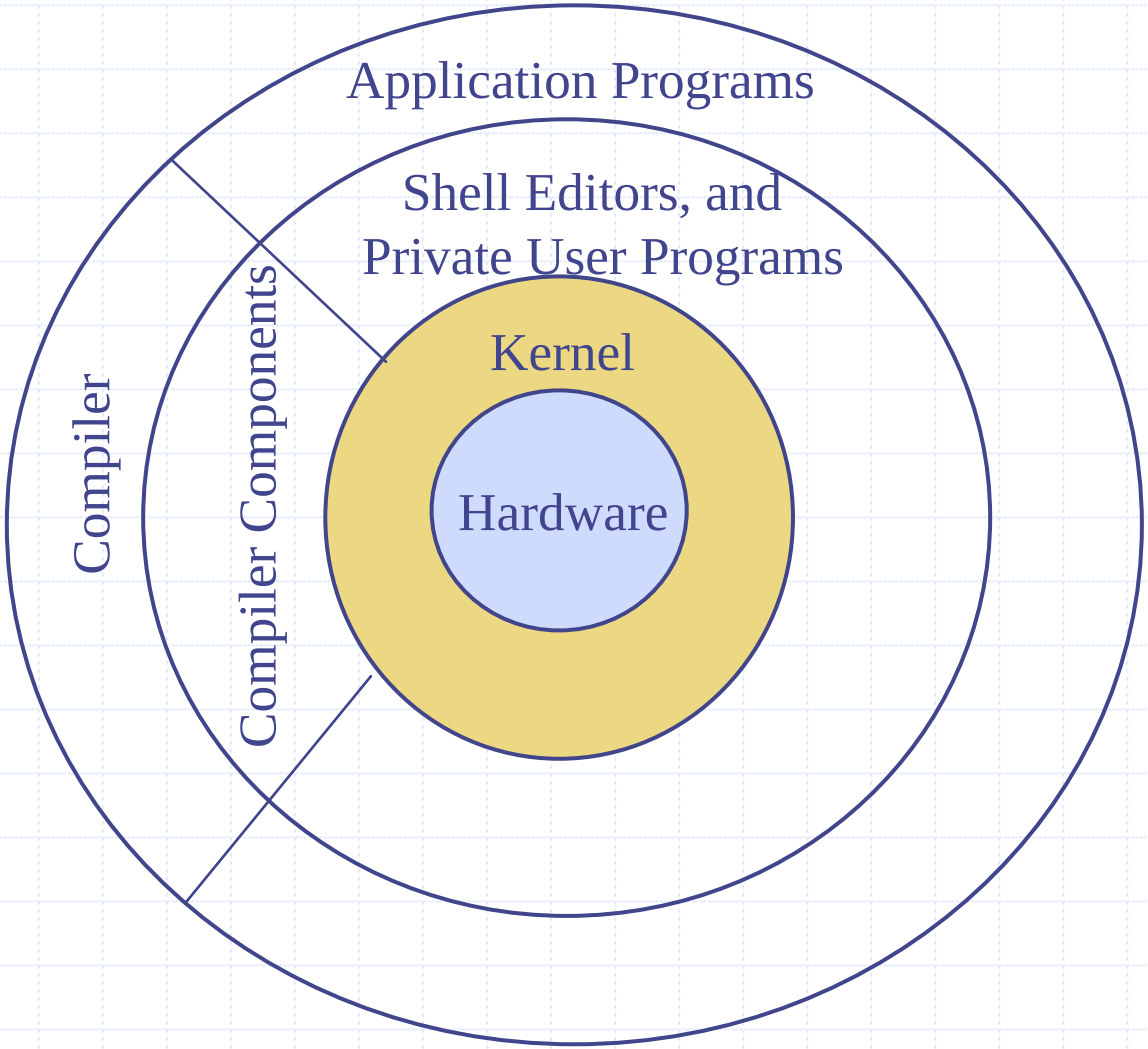
Operating System Design Hierarchie

Level	Name	Objekte	Beispieloperationen
13	Shell	User programming environment	Befehle aus shell Sprache
12	User processes	User processes	Quit, kill, suspend, resume
11	Directories	Directories	Create, destroy, attach, detach, search, list
10	Devices	Externe Devices, z.B. Drucker, Displays und Tastaturen	Create, destroy, open, close, read, write
9	File system	Files	Create, destroy, open, close read, write
8	Communications	Pipes	Create, destroy, open. close, read, write

Operating System Design Hierarchie

Level	Name	Objekte	Beispieloperationen
7	Virtual Memory	Segments, pages	Read, write, fetch
6	Local secondary store	Blocks of data, device channels	Read, write, allocate, free
5	Primitive processes	Primitive process, semaphores, ready list	Suspend, resume, wait, signal
4	Interrupts	Interrupt-handling programs	Invoke, mask, unmask, retry
3	Procedures	Procedures, call stack	Mark stack, call, return
2	Instruction Set	Evaluation stack, microprogram interpreter, scalar and array data	Load, store, add, subtract branch
1	Electronic circuits	Registers, gates, buses, etc.	Clear, transfer, activate, complement

UNIX Architektur



Zusammenfassung

- Ein Betriebssystem - Was ist das ?
- Aufgaben und Anforderungen
  - Herstellung einer komfortabel zu bedienenden, homogenen, virtuellen Maschine unter den Randbedingungen
    - Effizienz
    - einfacher Wartbarkeit / Weiterentwicklung
- einige Beispielarchitekturen für Betriebssysteme
  - Monolithisches System, Monolithischer Kernel, Schichtenmodell (Unix), Mikrokernelarchitektur (Windows 2000)

Mit dieser Vorlesung sollten Sie

- verstehen was ein Betriebssystem ist,
- einen groben Überblick über die vielfältigen Aufgaben bekommen

Wichtige Begriffe

mit denen wir uns in Zukunft noch genauer auseinander setzen werden

Prozeß, Prozeßimage, Adreßraum, Scheduling, virtueller Speicher, Dateisystem, E/A Gerät, Sicherheit, Client/Server Architektur, …