Betriebssysteme

Prof. Dr.-Ing. Tim Tiedemann

(http://www.informatik.haw-hamburg.de/~tiedemann)

→ EMIL „Betriebssysteme (Tdm) W16“

(basierend auf den Folien von Prof. Dr.-Ing. Martin Hübner)

Einbettung

- **Schichten, Anwendung**
- **Lehrveranstaltungen**

Motivation

- **Warum ist Betriebssystemwissen wichtig?**
  - Die Lösung komplexer Probleme (in einer Anwendung) verlangt oftmals Systemwissen, insbesondere BS-Wissen.
  - Ein BS bildet ein großes SW-System, daher sind Architektur und Techniken eines BS oft übertragbar.
  - Ein BS findet man nicht nur in „konventionellen“ Rechnern (PCs, Server, Mainframe) sondern überall – von der Anlagensteuerung über Flugzeug und Auto bis zum Handy und der Chipkarte.
- **Das wissen Sie am Ende der Vorlesung:**
  - Aufbau eines modernen BS
  - Konzepte eines BS und ihre Auswirkungen für Anwendungen
  - Algorithmen und Strategien zur effizienten Verwaltung und fairen Vergabe von Betriebsmitteln
  - **Kein Thema:** Wie bediene/konfiguriere/optimiere ich Windows?

Gliederung der Vorlesung (1)

1. Einführung & Überblick

1. Was ist ein Betriebssystem?
2. Überblick UNIX
3. Überblick Windows
4. Grundlegende Hardware-Konzepte
5. Die Struktur von Betriebssystemen

2. Prozesse

1. Das Prozessmodell
2. Das Threadmodell
3. Prozess-Scheduling

Gliederung der Vorlesung (2)

3. Prozess-Synchronisation

1. Einführung und Grundlagen
2. Aktives Warten
3. Semaphore
4. Monitore
5. Prozess-Synchronisation in UNIX
6. Prozess-Synchronisation in Windows
7. Deadlocks

4. Hauptspeicher-Verwaltung

1. Anforderungen und Grundlagen
2. Virtueller Speicher

Gliederung der Vorlesung (3)

5. Externe Geräte & Dateisysteme

1. Externe Geräte
2. Dateisysteme
3. Zuverlässigkeit von Dateisystemen

Literaturempfehlungen: Basisliteratur

- Andrew S. Tanenbaum: Modern Operating Systems, 4. Auflage, Pearson, 2015 [AT]
  Umfassendes Lehrbuch mit detaillierten Erklärungen inkl. Linux, Android und Windows
- Peter Mandl: Grundkurs Betriebssysteme, 2. Auflage, Vieweg+Teubner, 2010 [PM]
  Gut lesbares Lehrbuch auf Grundlage einer BS-Vorlesung an der FH München (pdf-Version zum Download im Pub verfügbar)
- Eduard Glatz: Betriebssysteme: Grundlagen, Konzepte, Systemprogrammierung, 2. Auflage, dpunkt Verlag, 2010 [EG]
  Umfassendes, anwendungsorientiertes Lehrbuch inkl. Beschreibung der Unix-Skriptprogrammierung
- Abraham Silberschatz, Peter Galvin, Greg Gagne: Operating System Concepts with Java, 8. Auflage, John Wiley & Sons, 2011 [SGG]
  Gut lesbares Standardwerk mit vielen Beispielen inkl. Linux und Windows
- Carsten Vogt: Betriebssysteme, Spektrum Akadem. Verlag, 2001 [CV]
  Leicht lesbares Buch mit guten Beispielen und einer Einführung in die UNIX-C-Schnittstelle

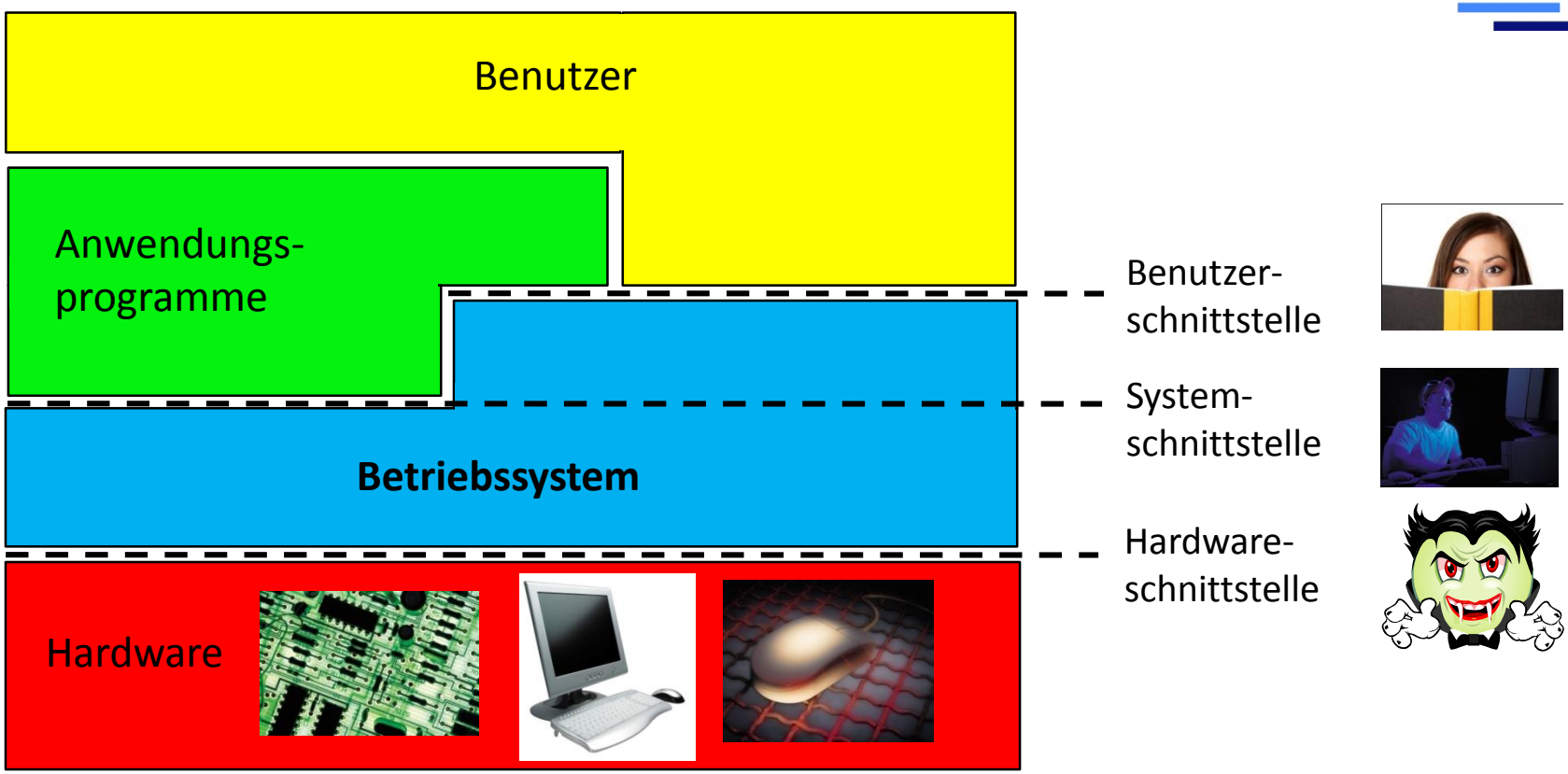
Literaturempfehlungen: Spezielle Vertiefungen

- Claudia Eckert: IT-Sicherheit, 8. Auflage, Oldenbourg Verlag, 2013
  Umfassende Darstellung aller Bereiche der IT-Sicherheit
- David A. Solomon, Mark Russinovich:
  Inside Microsoft Windows 2000, Microsoft Press, 2000
  Umfassende Beschreibung der Windows-Implementierung
  Windows-Tools: www.sysinternals.com
- Daniel B. Bovet, Marco Cesati: Understanding the Linux Kernel, 3. Auflage, O´Reilly, 2006
  Gute Einführung in die Grundkonzepte von Linux
- Paul Herrmann, Udo Kebschull, Wilhelm G. Spruth: Einführung in z/OS und OS/390, Oldenbourg Verlag, 2003
  Überblick über IBM-Mainframetechnologie inkl. JAVA-Anbindung
- ...

Kapitel 1
Einführung & Überblick

1. Was ist ein Betriebssystem?
2. Überblick UNIX
3. Überblick Windows
4. Grundlegende Hardware-Konzepte
5. Die Struktur von Betriebssystemen

Was ist ein Betriebssystem?



Ein Betriebssystem

- … bietet Schnittstellen zwischen dem Benutzer, den Anwendungsprogrammen und der Hardware
- … steuert die Ausführung von Programmen

Sichten auf ein Betriebssystem (1)

Benutzer / Programmierer-Sicht: **„Abstrakte Maschine“**

- **Hauptziel: Einfache und komfortable Schnittstellen!**
  - Benutzerschnittstelle
    - Graphiksystem (GUI)
    - Kommandointerpreter mit Skript-Sprache
  - Programmierschnittstelle
    - Hardware-unabhängige Prozeduraufrufe („System Calls“) für alle Betriebssystemfunktionen

Sichten auf ein Betriebssystem (2)

System-Sicht: **„Betriebsmittelverwalter“**

- **Hauptziel: Effiziente Betriebsmittelausnutzung!**
  - Prozessor
  - Speicher
  - Dateien
  - Ein-/Ausgabegeräte
- Ein Betriebssystem verwaltet die Betriebsmittel und teilt sie den Anwenderprogrammen zu

Betriebssystem – Generationen und parallele Hardware-Entwicklungen

- Erste Generation 1945 - 1955
  - Relais, Elektronenröhren // Schalttafeln
- Zweite Generation 1955 - 1965
  - Transistoren // Stapelverarbeitung (“batch systems”)
- Dritte Generation 1965 – 1980
  - VLSI (“Very Large Scale Integration”) // Multiprogramming
- Vierte Generation 1980 – heute
  - Personal Computer // Mobile Computer // Verteilte Systeme

Erste Generation: 1945 - 1955

- Kein Betriebssystem!
  - Nur ein Benutzer und ein Programm gleichzeitig
  - Programmierer = Operator = Benutzer („Open Shop“)
  - Computer werden von einer Konsole bedient
    - Mit Lampen zur Statusanzeige
    - Mit Schaltern für bitweise Eingabe (z.B. Startadresse)
    - mit Eingabegeräten wie Lochstreifenlesern etc.
    - mit Ausgabegeräten wie Drucker etc.
  - Zeitbuchung auf Papier
  - Programmausführung war umständlich
  - Einzige Unterstützung bei der Fehlersuche: Speicherauszug („Core Dump“)

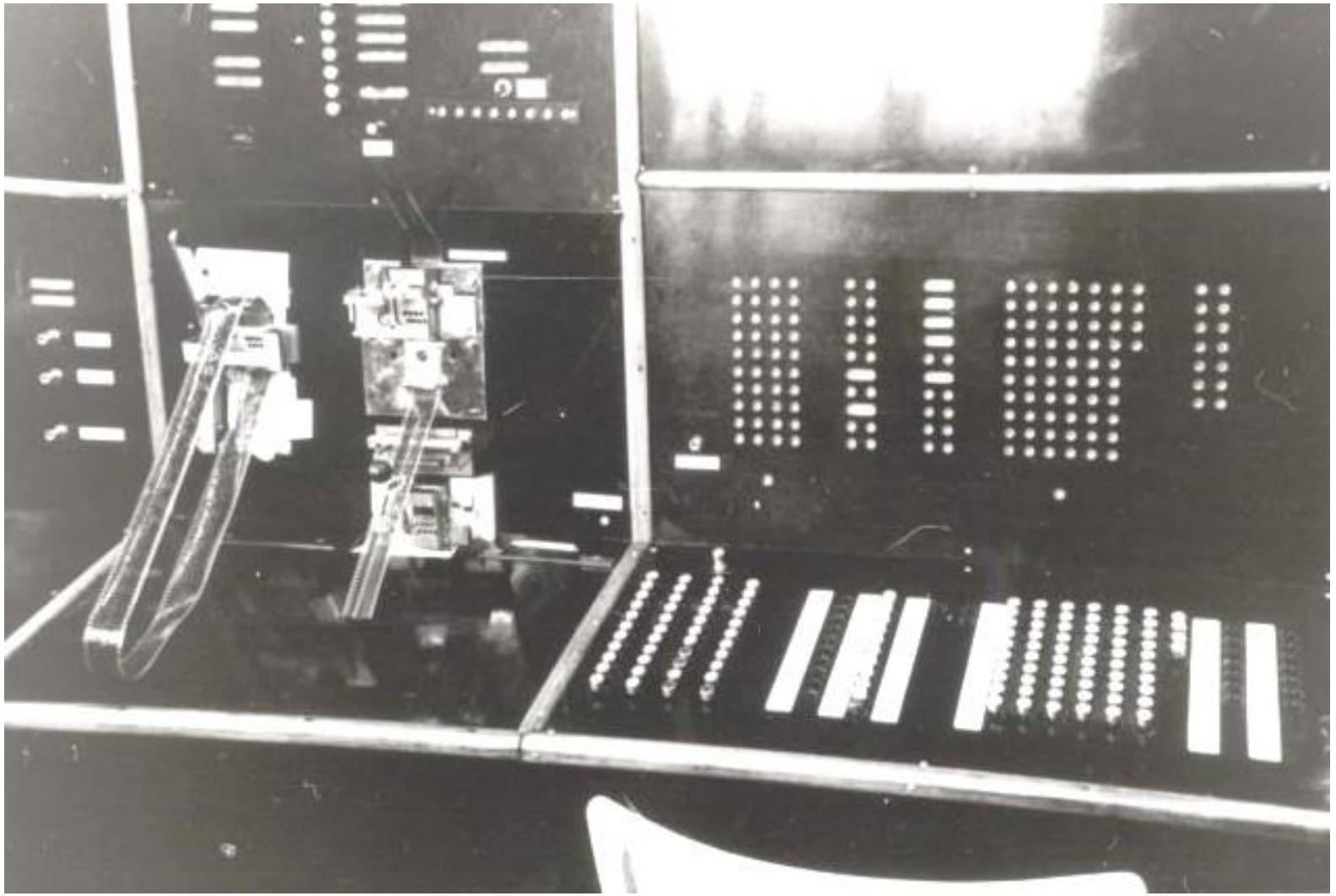
Väter:

John von Neumann (USA - Princeton)

Konrad Zuse (Deutschland)

Howard Aiken (USA - Harvard)

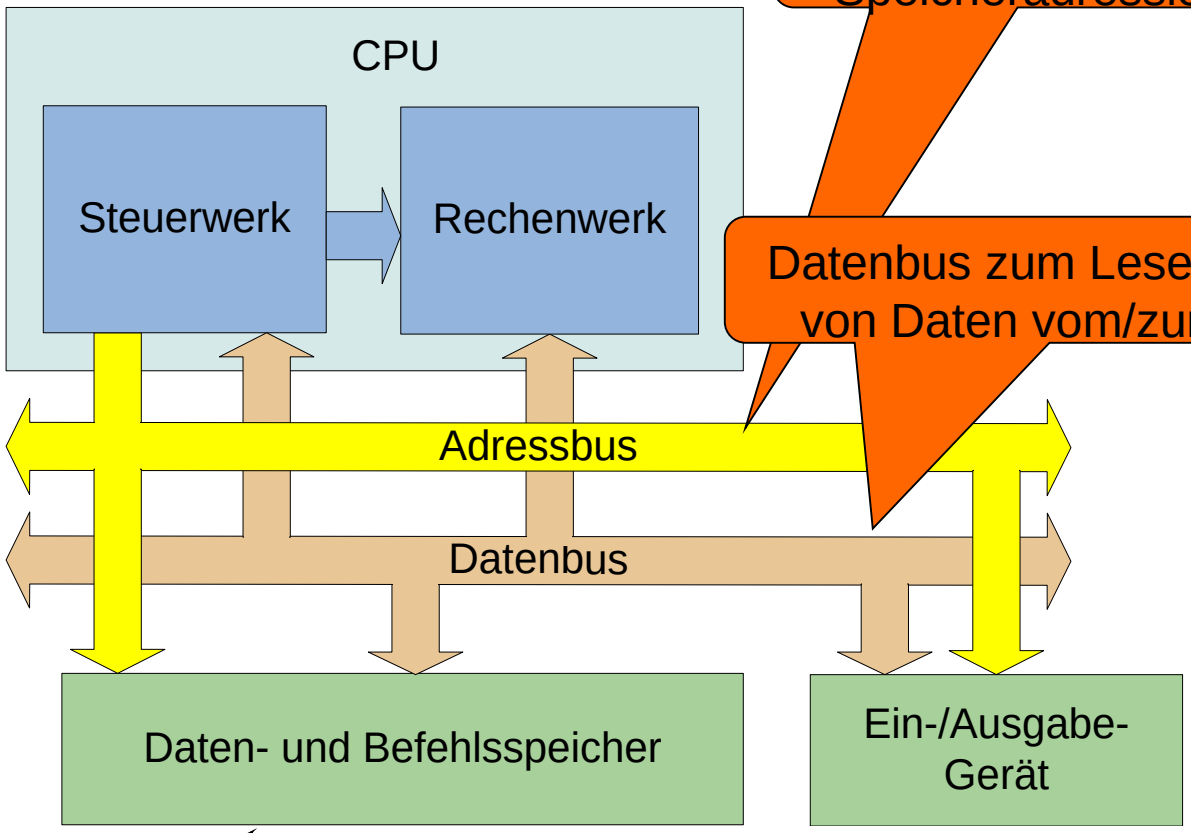
Schalttafel einer Zuse Z4 (1945)



„Von-Neumann-Architektur“

Fetch-Decode-Execute Cycle

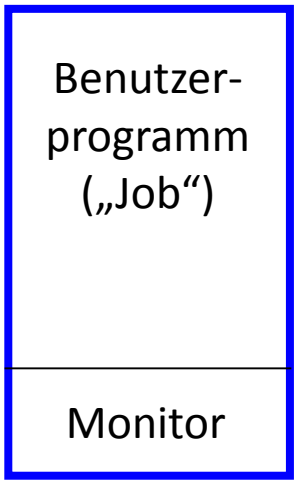
Adressbus zur Speicheradressierung



Datenbus zum Lesen/ Schreiben von Daten vom/zum Speicher

Ein Speicher für Befehle und Daten

Zweite Generation: 1955 - 1965

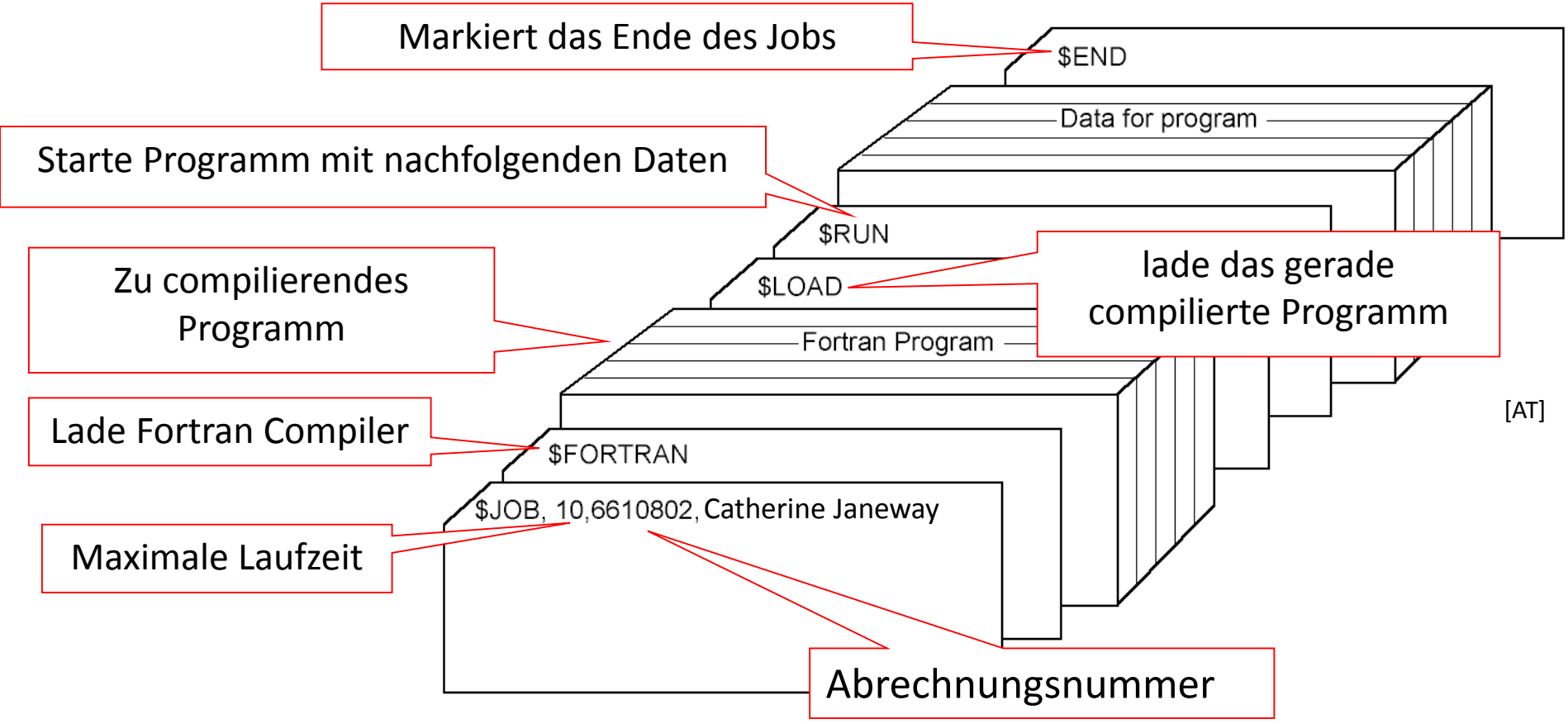


- Einfache Batch-Systeme („Stapelverarbeitung“)
  - Monitor (Betriebssystemvorläufer)
    - Software, die den Ablauf der Programme steuert
    - Der Monitor ist speicherresident, d.h. er ist immer geladen
    - Der Monitor lädt ein Programm und die Eingabedaten in den Hauptspeicher (von Lochkarte / Band) und verzweigt zu dessen Startadresse
    - Nach Programmende / bei Programmfehler wird wieder der Monitor aufgerufen, der das nächste Programm lädt
  - Computer wurden im Rechenzentrum betrieben
    - Zugang für Benutzer gesperrt
    - Benutzer gab sein Programm (auf Lochkarten) ab
    - Benutzer erhielt Ergebnis z.B. durch einen Ausdruck später

Job Control Language (JCL)

- Eine Kommmandosprache
  - entwickelt, um dem Monitor für das gerade aktuelle Programm („Job“) Informationen geben zu können
  - gibt Kommandos an den Monitor:
    - welcher Compiler genutzt werden soll
    - welche Daten genommen werden sollen
    - welche Ein- / Ausgabe gewählt werden soll
    - etc.
  - JCL wurde im Laufe der Entwicklung immer leistungsfähiger

Struktur eines Fortran-Jobs (Lochkarten)



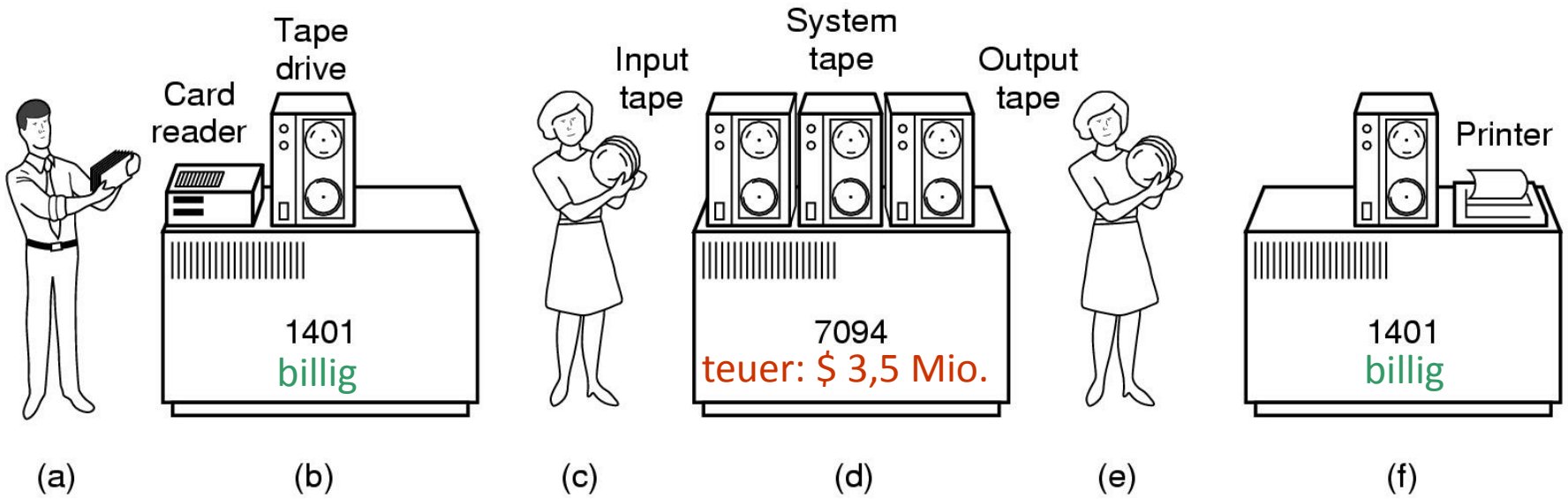
[AT]

Uniprogramming

- Nur ein Programm gleichzeitig!
- Prozessor wartet auf das Ende jeder (langsamen!) I/O-Operation, bevor das Programm fortfahren kann.

(„idle“: im Leerlauf)

Optimierte Stapelverarbeitung



[AT]

- Programmierer bringt Lochkarten zur IBM 1401
- Karteninhalt wird auf Band geschrieben
- Sammeln von Jobs; anschließend Zurückspulen des Bandes
- Band wird in die IBM 7094 eingelegt (Hauptrechner); Jobabarbeitung
- Druckoutput wird wieder auf Band geschrieben
- Ausdruck vom Band über die IBM 1401

Vergleich Kartenleser / Bandstation

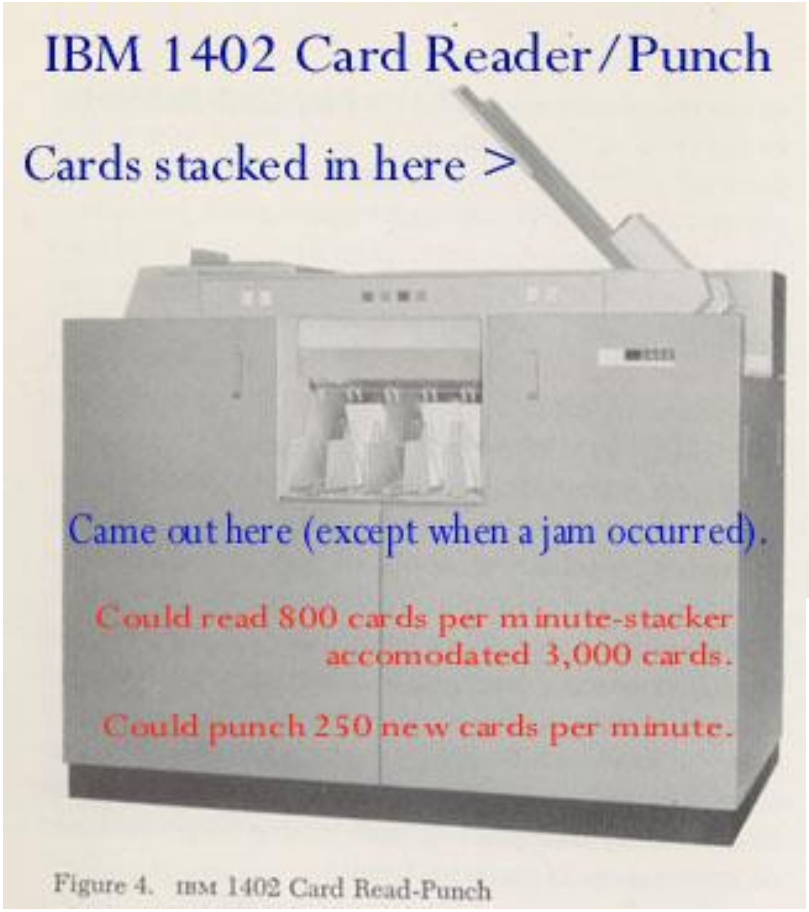


Figure 4. IBM 1402 Card Read-Punch

OPERATING CHARACTERISTICS	729-II	729-IV	7330
Density, Characters Per Inch	200 or 556	200 or 556	200 or 556
Tape Speed, Inches Per Second	75	112.5	36
Inter-Record Gap Size, Inches	3/4	3/4	3/4
Character Rate, Characters Per Second	15,000 or 41,667	22,500 or 62,500	7,200 or 20,016
High Speed Rewind, Minutes	1.2	.9	2.2
Regular Rewind, Inches Per Second	75	112.5	36

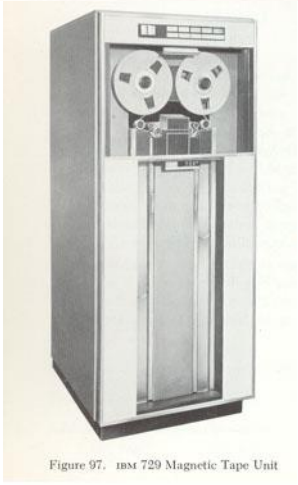


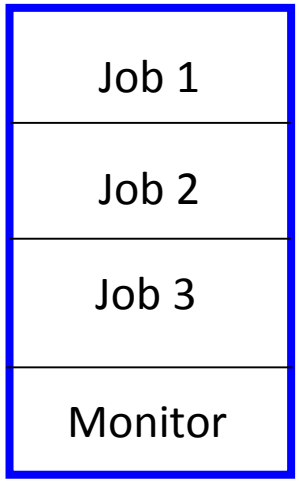
Figure 97. IBM 729 Magnetic Tape Unit

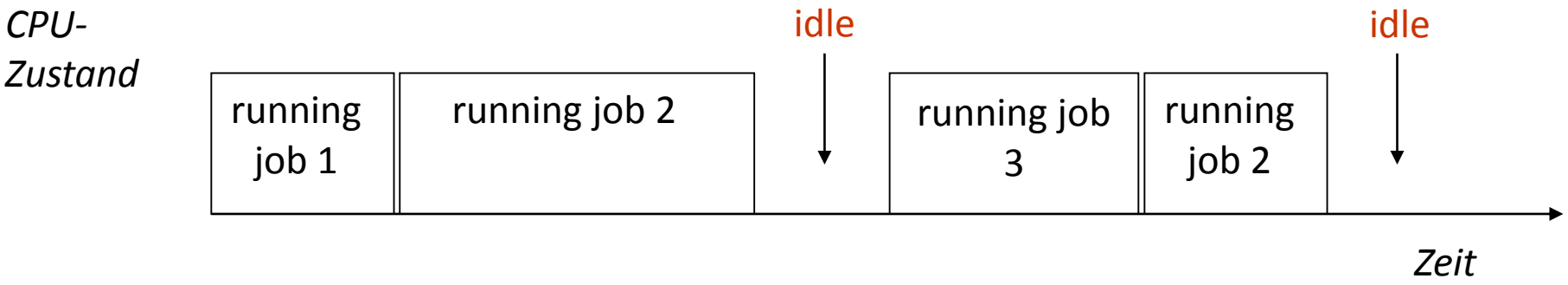
Dritte Generation: 1965 - 1980

- Die Hardware-Hersteller (IBM: OS/360) lernten schnell, dass folgende Hardware-Eigenschaften für ein stabiles, sauber strukturiertes Betriebssystem notwendig sind:
  - Speicherschutz
    - Das Benutzerprogramm darf das Monitor-Programm und Monitor-Datenbereich nicht verändern
  - Timer / Interrupts
    - erlauben es, regelmäßig aus dem Benutzerprogramm zum Monitor zurückzukehren
    - verhindern, das ein Programm die CPU nicht wieder abgibt (d.h. den Rechner monopolisiert)
  - Privilegierte Operationen
    - Nur Monitor soll z.B. I/O-Befehle ausführen können
    - 2 Modi-Betrieb notwendig (User-/Kernmodus)

Multiprogramming

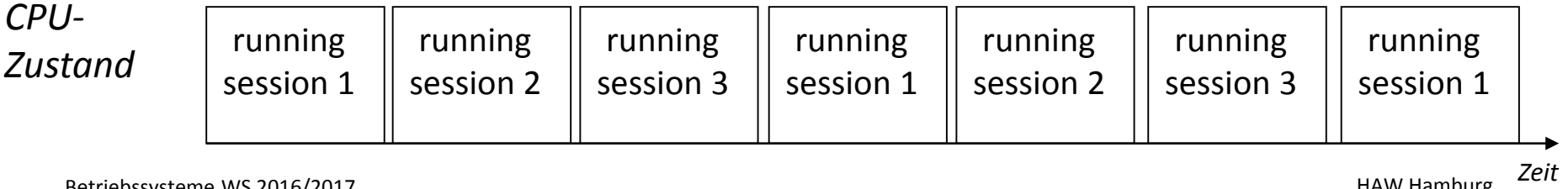
- Mehrere Programme (Jobs) sind gleichzeitig im Hauptspeicher
- Wenn ein Programm (Job) die CPU „freiwillig“ abgibt und auf I/O wartet, kann der Monitor die CPU auf ein anderes Programm umschalten
  (Welches? → Scheduling!)





Weiterentwicklung: Time Sharing

- Wunsch nach Interaktion mit dem Computer führt zur Entwicklung von Time-Sharing Systemen
- Benutzt Multiprogramming, um viele interaktive Jobs („Sessions“) zu verarbeiten
- Die CPU-Zeit wird zwischen den vielen Programmen / Benutzern aufgeteilt:
  - Unterbrechung eines Programms nach Ablauf einer (kurzen) „Zeitscheibe“ (durch Timer und Interrupt)
- Die Benutzer haben durch viele Bildschirme gleichzeitig Zugriff auf das System
- Jeder Benutzer kann wie an einem Einplatz-System arbeiten



Vergleich zwischen Batch Multiprogramming und Time Sharing

- Mit den unterschiedlichen Betriebsarten werden unterschiedliche Ziele verfolgt :

	Batch Multiprogramming	Time Sharing
Prinzipielles Ziel	**Maximierung der CPU-Auslastung (Durchsatz)**	**Minimierung der Antwortzeit**
Anweisungen kommen von:	JCL – Elementen, die jedem Programm beigefügt sind	Kommandos, die am Bildschirm eingegeben werden
Unterbrechung wegen:	I/O-Anforderung	Ablauf der Zeitscheibe

Vierte Generation 1980 – heute

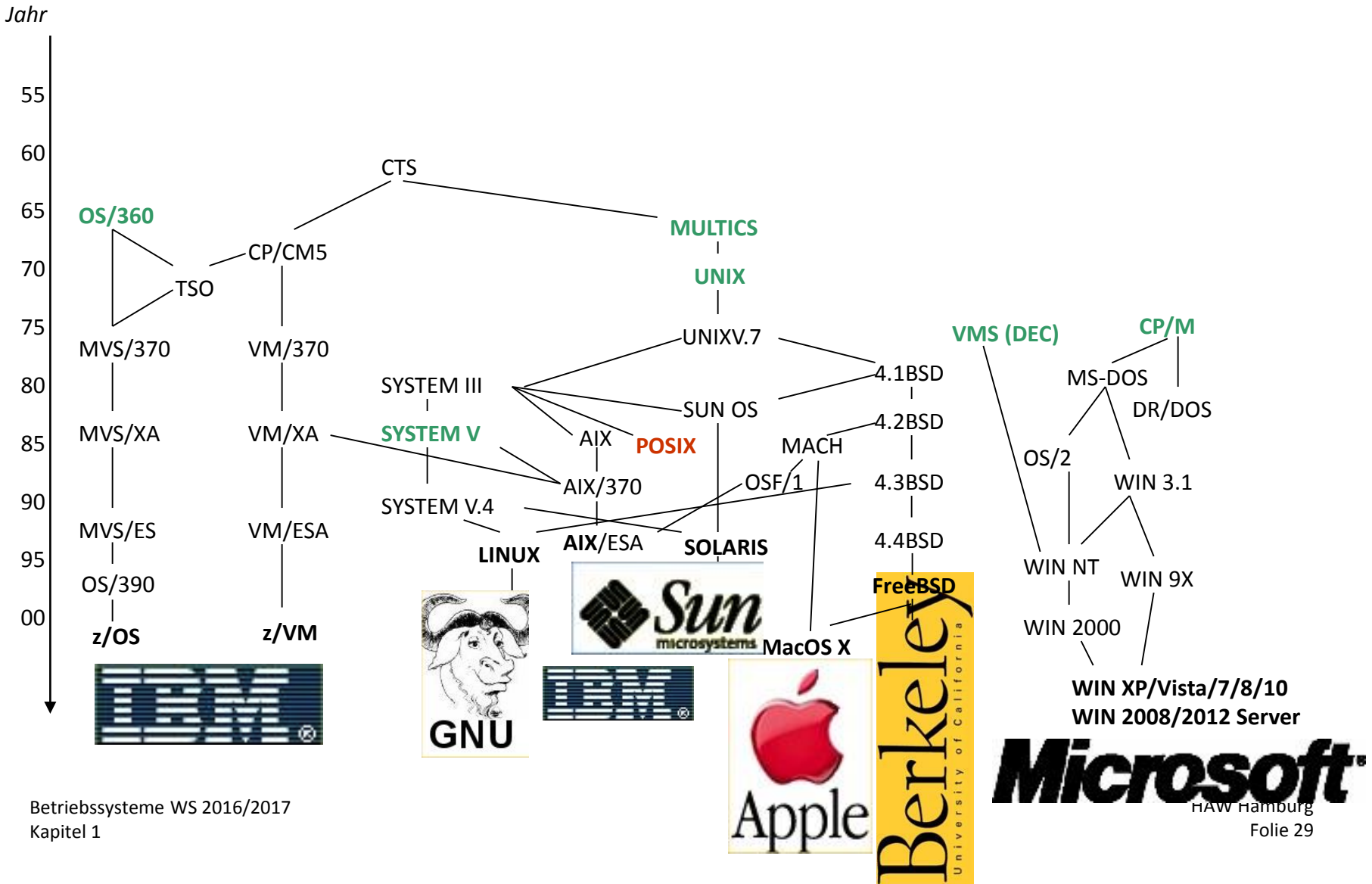
- Multiprozessorsysteme
  - Meist Symmetrisches Multiprocessing (SMP):
    Der BS-Code ist nur einmal im Speicher, doch jede CPU kann ihn ausführen (→ Intelligentes Sperren nötig!)
- Multicomputersysteme
  - Cluster
  - Storage Area Networks
- Mobile Systeme
  - Handy / PDA / Smartphone
- Verteilte Systeme
  - Client-/Server-Systeme
  - Netzbetriebssysteme
  - Verzeichnisdienste
  - …

➔ Bauen auf den Erfahrungen und Prinzipien der 1.-3. Generation auf!

Aktuelle Betriebssystem - Typen

- Mainframe-Betriebssystem
  - Sehr hohe I/O-Kapazität, sehr viele parallele Programme *(IBM z/OS)*
- Server-Betriebssystem
  - Hohe I/O-Kapazität, User-Interface nicht wichtig *(Windows Server, Unix)*
- Echtzeit-Betriebssystem
  - Zeitkritische Anwendungen, z.B. Anlagensteuerung *(VxWorks, QNX)*
- Desktop-Betriebssystem
  - Workstation, Notebook *(Windows, Linux, Mac OS-X)*
- Handheld-Betriebssystem
  - Handy, Smartphone, Tablet *(Android (Linux), iPhone iOS, Windows Phone)* - z.T. „abgespeckte“ Desktop-Systeme
- Eingebettetes Betriebssystem („Embedded System“)
  - BS in technischen Geräten, z.B. Waschmaschinen, Fernsehern, Chipkarten

Betriebssysteme - Entwicklung



Zusammenfassung Abschnitt 1: Was ist ein Betriebssystem?

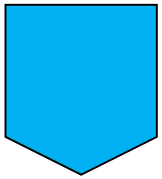
- Schnittstelle zwischen Hardware und Applikationen
- Mögliche Sichten:
  - „Virtuelle Maschine“
  - Betriebsmittelverwalter
- Vier Generationen (1945 – heute)
- Permanente Orientierung an den Möglichkeiten der Hardware
- Die modernen Konzepte haben sich im Lauf der Zeit entwickelt
- **`Betriebssystem != Betriebssystem`**

Kapitel 1
Einführung & Überblick

1. Was ist ein Betriebssystem?
2. Überblick UNIX
3. Überblick Windows
4. Grundlegende Hardware-Konzepte
5. Die Struktur von Betriebssystemen

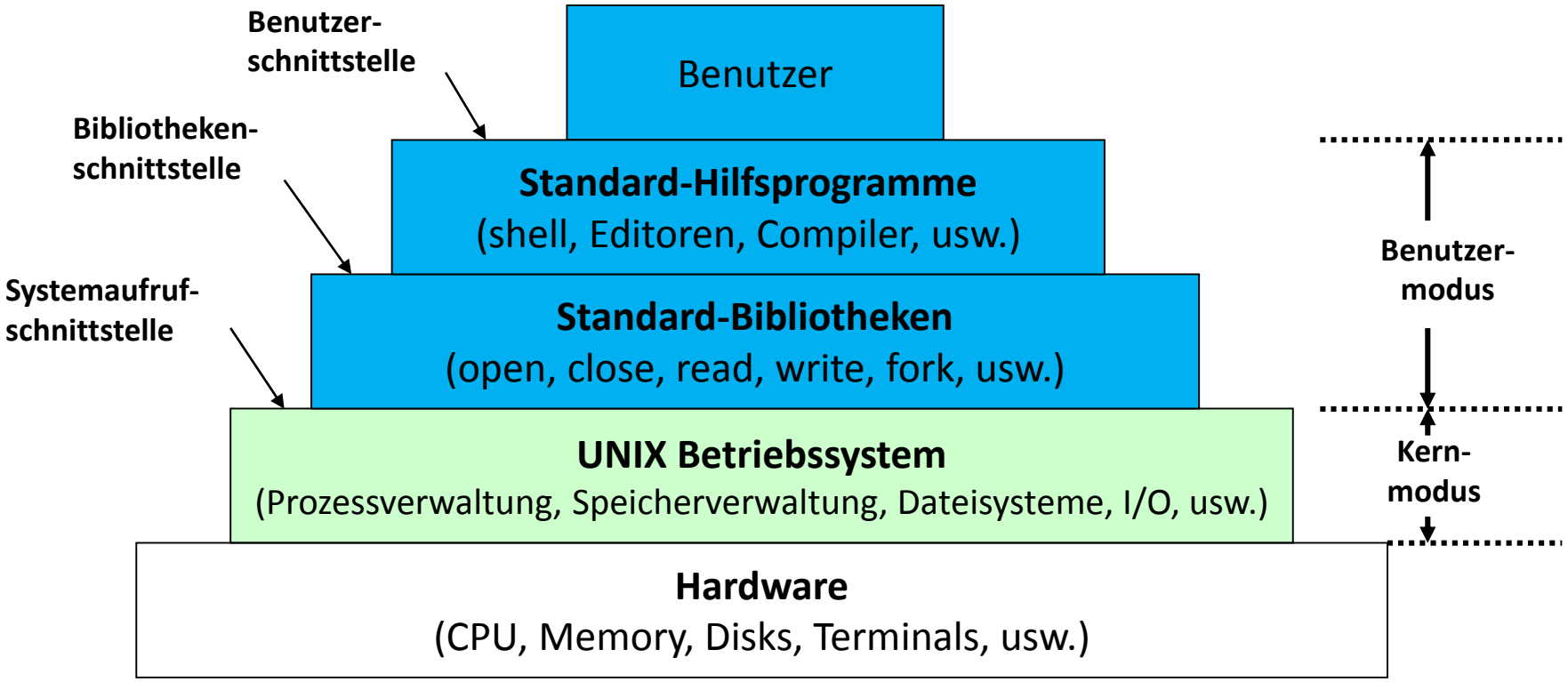
UNIX - Ziele

- Von Programmierern **für Programmierer** entworfen
- Mehrheit der **Benutzer ist relativ erfahren** und oftmals in komplexe Softwareentwicklung eingebunden
- Modell „**enge Zusammenarbeit von Programmierer-Team**“ (unterscheidet sich fundamental vom Modell „persönlicher Computer mit einzigem Anfänger“)



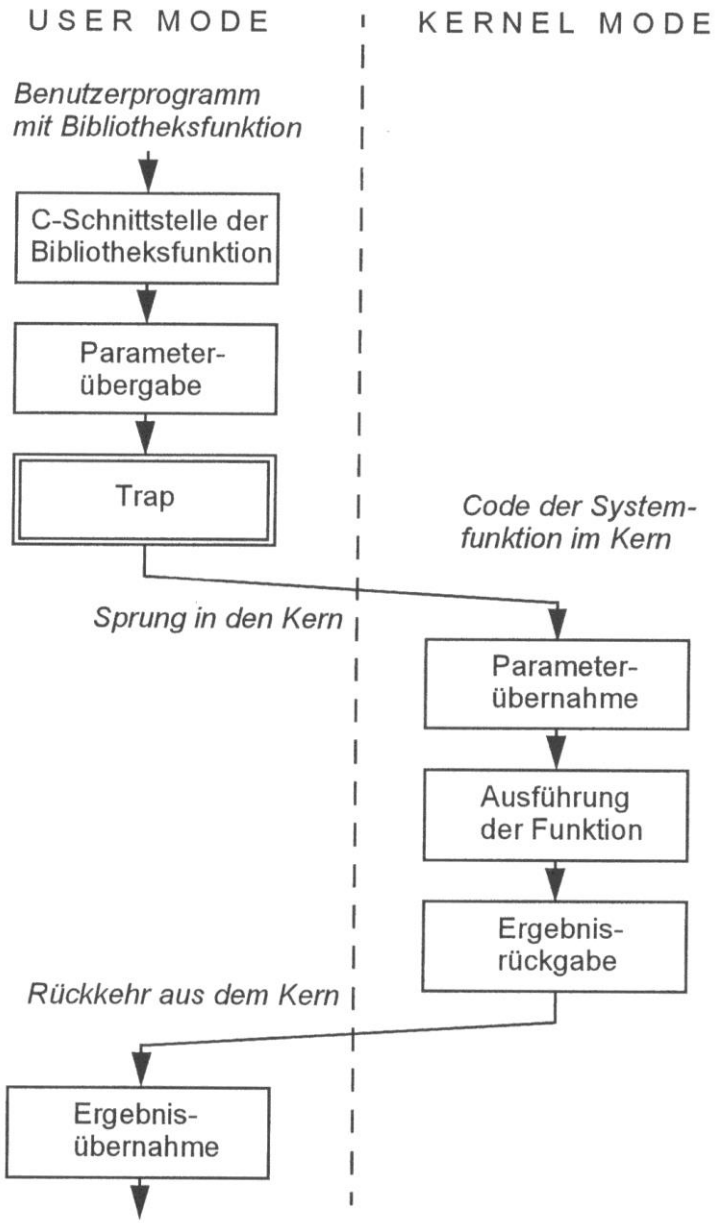
Die Frage „**Was erwartet ein guter Programmierer vom System?**“ ist bei den wesentlichen Designentscheidungen von UNIX zu Grunde gelegt worden

UNIX – Schnittstellen



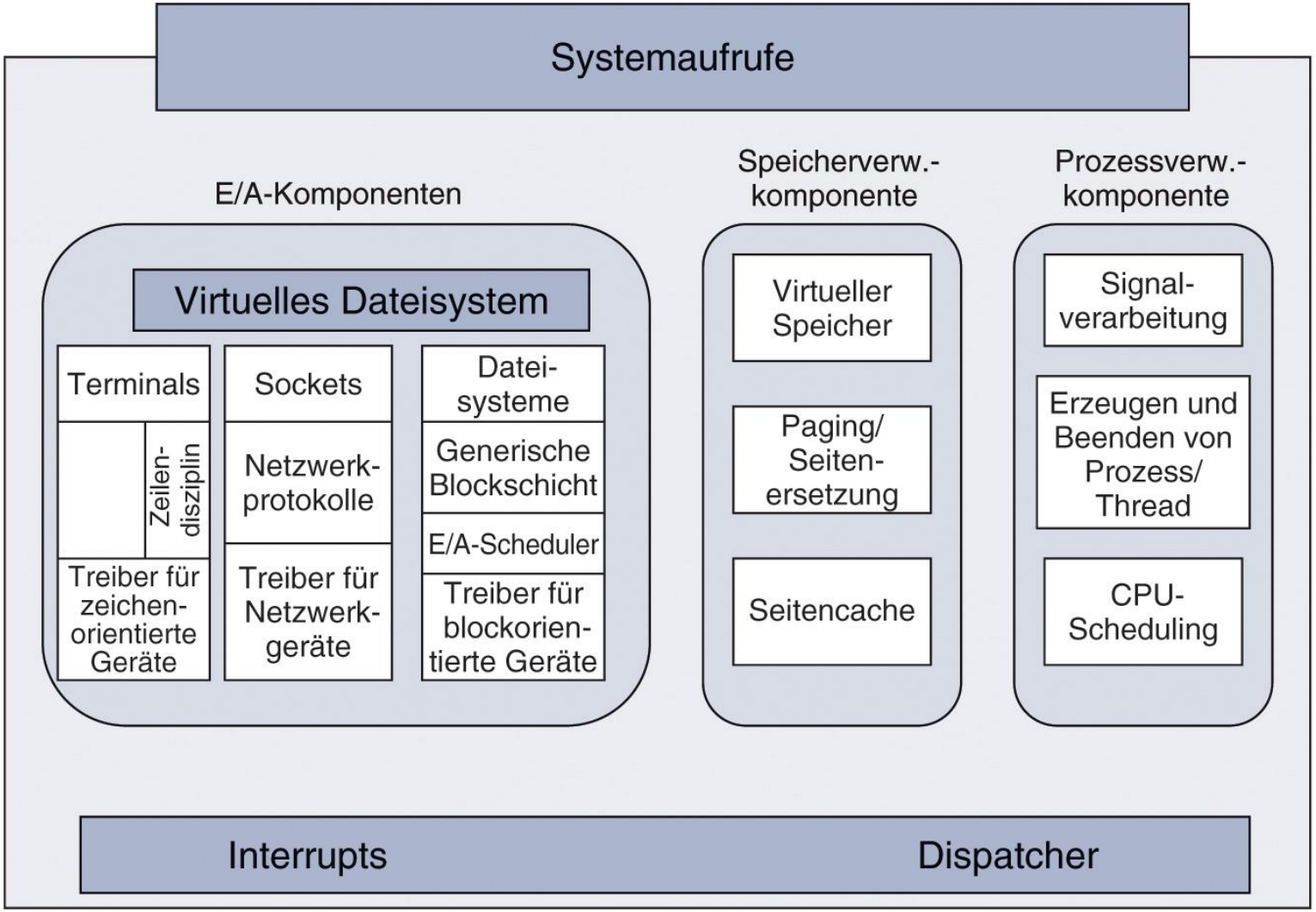
Systemaufrufe in UNIX

- Systemaufruf (Wechsel in den Kern mittels SW-Interrupt) nur über Bibliotheksfunktion möglich
- Einbindung in C-Programme über #include der Headerdateien (/usr/include bzw. /usr/include/sys)
- TRAP-Funktion verursacht (Software-)Interrupt mit Wechsel in den Kernel-Mode → „System Call“ *(Parameterübergabe mittels Stack)*



[CV]

UNIX – Architektur (Linux)



Dienstprogramme

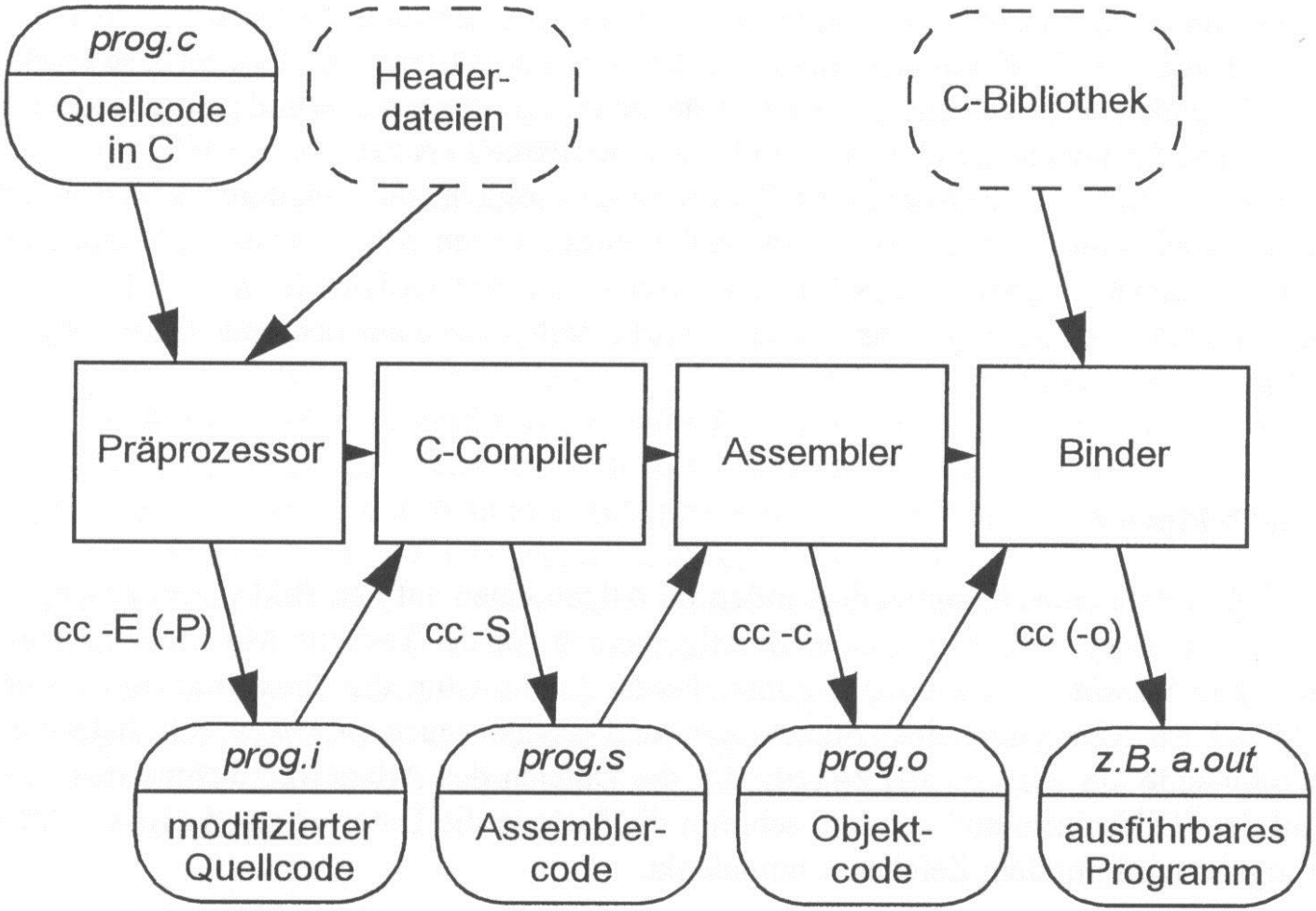
- Shell (Kommandointerpreter)
  - sh, csh, bsh, ksh, bash, tsh, …
  - Syntax:
    `befehl {-option} {argument} [&]`
- Editor
  - emacs, vi, …
- Dateien finden und ändern
  - grep, sed, …
- C-Compiler
  - cc, gcc, ..

Shell (Kommandointerpreter): Was geschieht bei einer Befehlsausführung?

- Der Benutzer gibt einen Befehl ein.
- Die Shell analysiert den Befehl:
  - Führt ihn selbst aus, falls es sich um einen „Built-in“-Befehl handelt
  - Lädt ansonsten das angegebene (System-)Programm und erzeugt damit einen neuen „Prozess“.
    - Der neue Prozess führt den Befehl aus.
    - Der neue Prozess liefert das Ergebnis an die Shell zurück und wird beendet (terminiert).
- Die Shell gibt das Ergebnis an den Benutzer weiter.
- Die Shell gibt ein neues Bereitzeichen aus.

Mehrere Befehle können in einer Datei übergeben werden → „Shell-Skript“

Schritte des UNIX-C-Compilers



[CV]

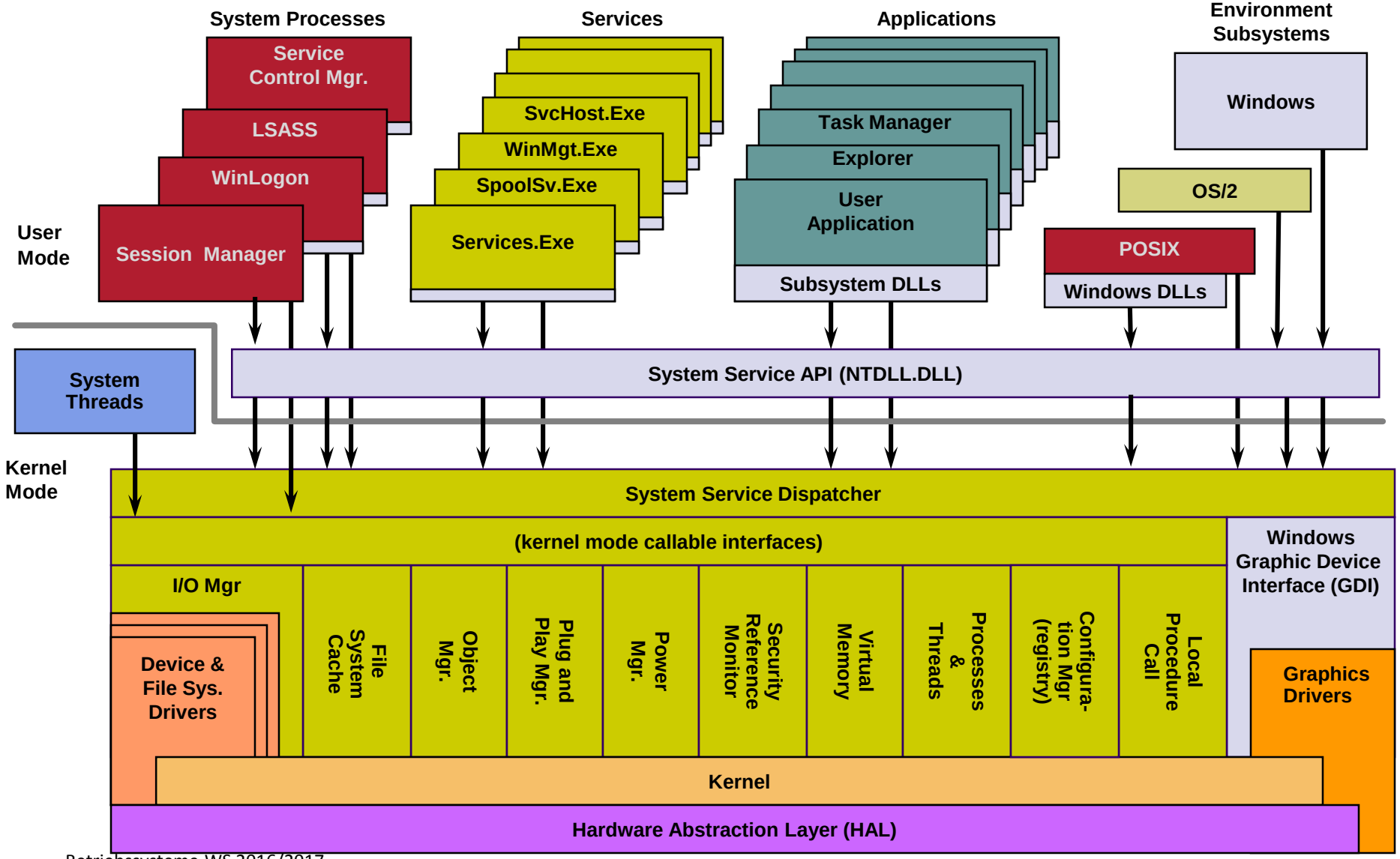
Kapitel 1
Einführung & Überblick

1. Was ist ein Betriebssystem?
2. Überblick UNIX
3. Überblick Windows
4. Grundlegende Hardware-Konzepte
5. Die Struktur von Betriebssystemen

Windows 2000/XP/Vista/7/8/10 - Struktur

- Modulare Struktur ➔ Flexibilität
- Überwiegend in C geschrieben (kleine Teile Assembler bzw. C++), dennoch Umsetzung eines Objektmodells
  ➔ Portabilität gegeben
- Modifizierter Microkernel
  - Keine reine Microkernel – Architektur, da gemeinsamer Adressraum aller Module
  - Viele Funktionen außerhalb des Kernels laufen im Kernel-Modus (Vermeidung von Kontextwechseln)
- Module können entfernt / aktualisiert / ersetzt werden, ohne das gesamte System zu ändern

Windows Architektur



hardware interfaces (buses, I/O devices, interrupts, interval timers, DMA, memory cache control, etc., etc.)