1. Foliensatz
Betriebssysteme und Rechnernetze

Prof. Dr. Christian Baun

Frankfurt University of Applied Sciences
(1971–2014: Fachhochschule Frankfurt am Main)
Fachbereich Informatik und Ingenieurwissenschaften
christianbaun@fb2.fra-uas.de

Organisatorisches zur Vorlesung und Übung

- **E-Mail:** `christianbaun@fb2.fra-uas.de`

!!! Sagen Sie mir frühzeitig wenn es Probleme gibt !!!

- **Homepage:** `http://www.christianbaun.de`

!!! Schauen Sie regelmäßig auf die Vorlesungsseite !!!

- Die Homepage enthält u.a. die **Vorlesungsunterlagen**
  - **Präsentationsfolien**
  - **Übungsblätter**
  - **Musterlösungen** der Übungsblätter
  - Alte **Klausuren** und deren **Musterlösungen**

Wie ist das Passwort?

Es gibt kein Passwort!

- Die regelmäßige Übungsteilname ist keine Voraussetzung zur Klausurteilnahme – sinnvoll ist die Teilnahme aber schon

Portfolioprüfung

- Das Modul Betriebssysteme und Netzwerke sieht nicht einfach eine Klausur als Prüfungsleistung vor, sondern eine **Portfolioprüfung**
- Das Portfolio besteht aus 2 Teilen („Werkstücken“)
  - **Werkstück A** ist eine praktische Übung
    - Diese müssen die Sie bis zum Ende des Semesters erfolgreich bearbeiten
    - Dafür müssen Sie im Team die Lösung für eine Aufgabe implementieren und eine Dokumentation schreiben und abgeben
    - Zudem müssen Sie ihre Lösung präsentieren
  - **Werkstück B** ist ein schriftliches Testat („Klausur“) über 60 Minuten
- Gewichtung der Leistungen aus den Werkstücken: 50:50
  - Zum Bestehen genügen 50% der erreichbaren Punkte

Werkstück A – Aufgaben (Alternativen), Partnersuche, Abgabe Lösungsskizze, finale Abgabe,...

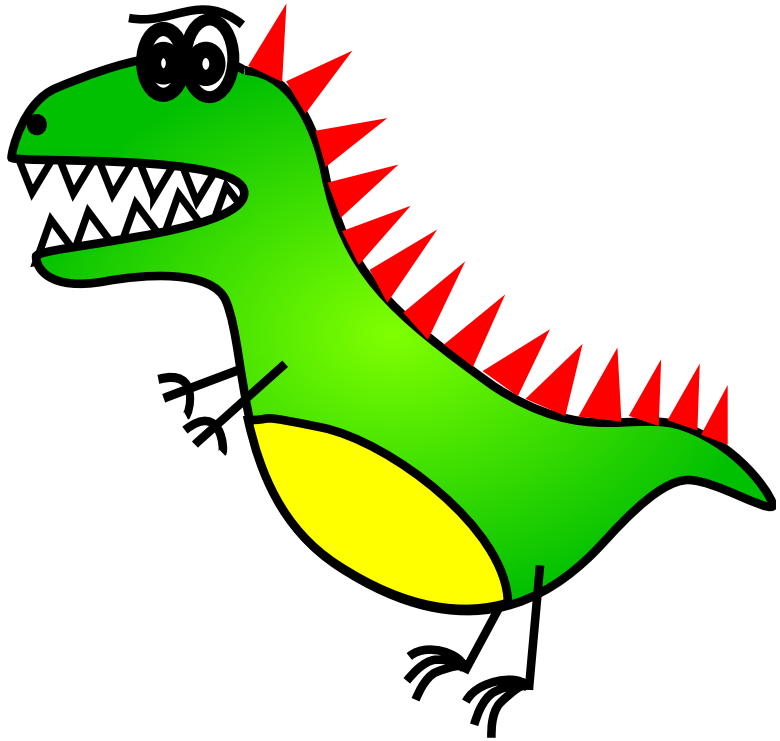
`https://campuas.frankfurt-university.de/course/view.php?id=3004`

!!! Achtung !!!

Ein „Mitnehmen“ der Punkte von Werkstück A in spätere Semester ist nicht möglich

Ausrichtung der Vorlesung BSRN

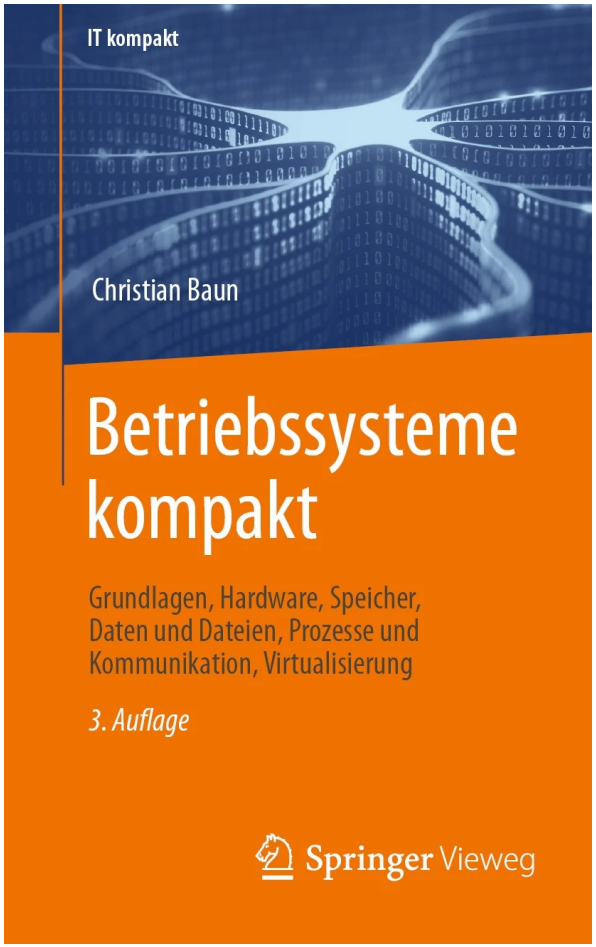
Bildquelle: http://pixabay.com (CC0)

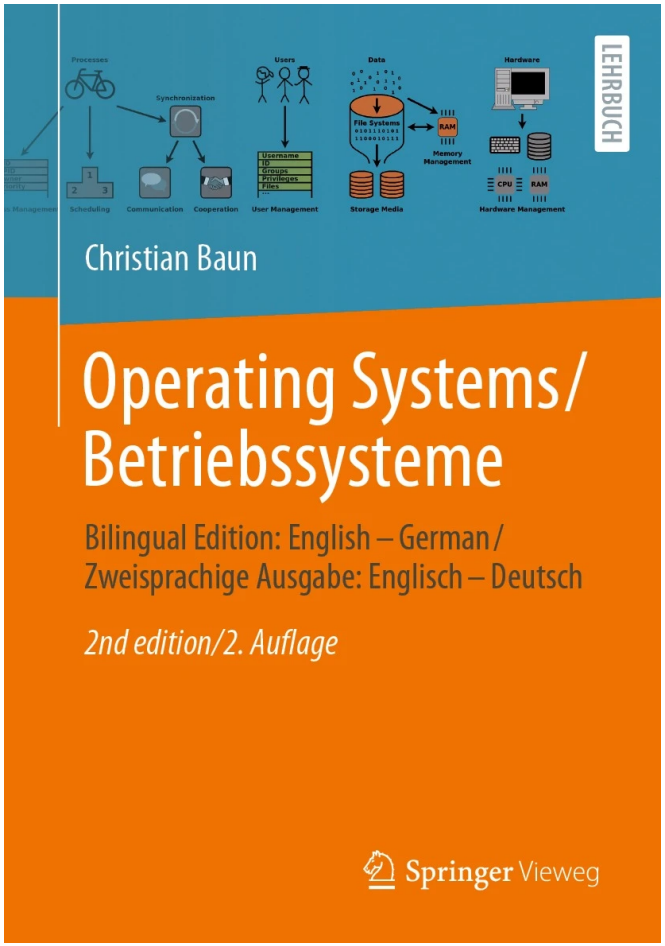


- Früher (bis SS2015)...
  - gab es 2 Vorlesungen:
    - **Betriebssysteme** (4 SWS)
    - **Rechnernetze** (4 SWS)
- Heute (seit SS2016)...
  - gibt es nur noch **Betriebssysteme und Rechnernetze** (4 SWS)

- Fakt: 50% der Inhalte von **Betriebssys.** und **Rechnern.** fallen weg
  - Das macht eine umfangreiche Ausbildung in den o. g. Themen unmöglich!
  - Was bleibt:
    - Einen Überblick über die Zusammenhänge vermitteln („das große Ganze“)
    - Einzelne Themen vertiefen – vieles andere streichen
    - Die Ausrichtung der Vorlesung BSRN ist ca. 60% BS und 40% RN

Literatur zum Thema Betriebssysteme





- Meine Vorlesungsunterlagen waren die Grundlage für diese Lehrbücher
- Das Layout des bilingualen Buches ist zweispaltig (Englisch/Deutsch)

Der Download beider Bücher ist kostenfrei aus dem Intranet über die Bibliothek der FRA-UAS möglich

Lernziele

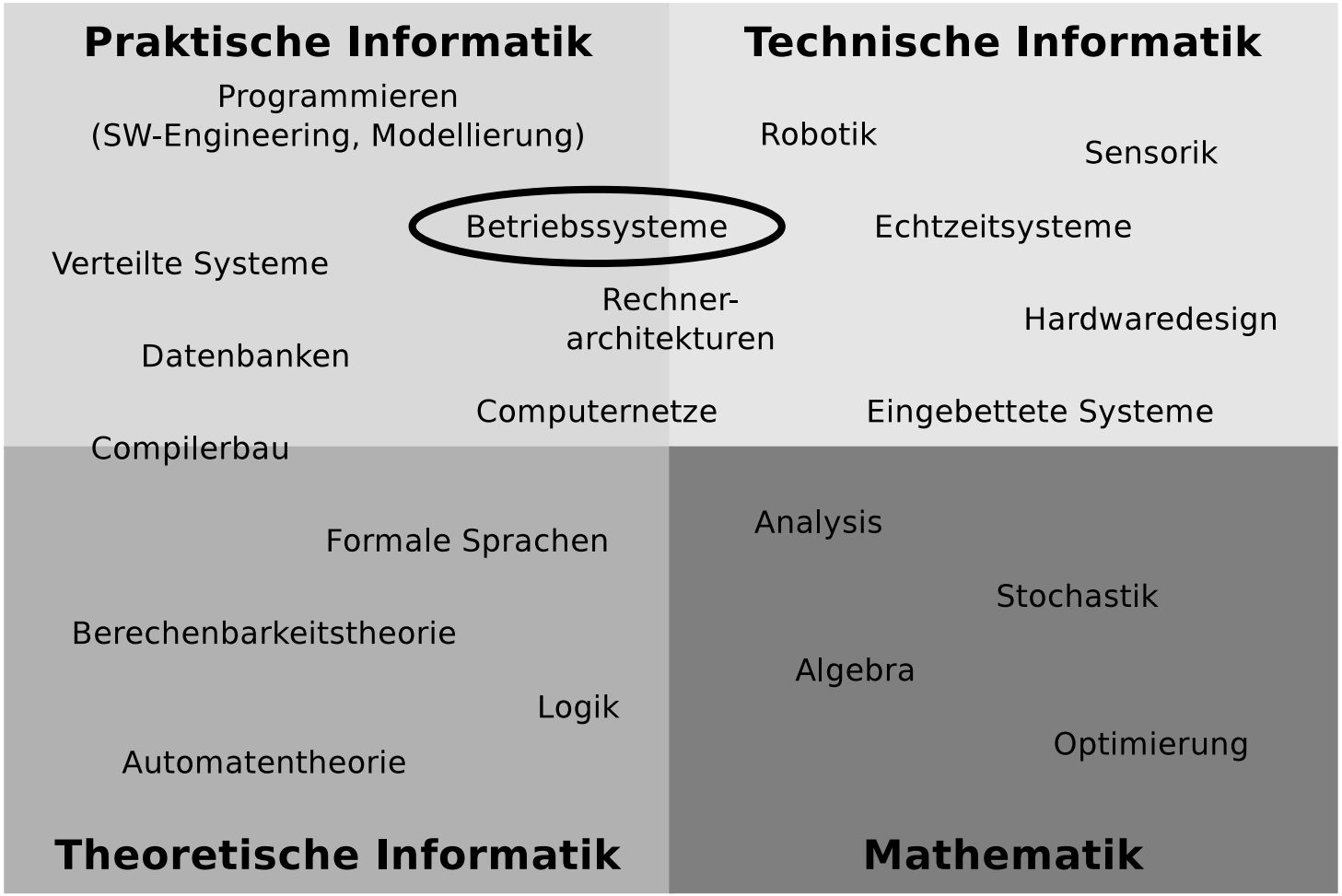
- Am Ende dieses Foliensatzes kennen/verstehen Sie...
  - wie die **Betriebssysteme** in die Informatik eingeordnet sind
  - wie die Entwicklungen der Hardware die **Entwicklung der Betriebssysteme** beeinflusst hat
    - Stapelverarbeitung
    - Einzelprogrammbetrieb (Singletasking)
    - Mehrprogrammbetrieb (Multitasking)
    - Dialogbetrieb (Time Sharing)
  - die **Kernfunktionalitäten** der Betriebssysteme:
    - Speicherverwaltung, Dateisysteme, Systemaufrufe, Prozessverwaltung, Interprozesskommunikation, Prozesssynchronisation
  - den **Betriebssystemaufbau** (unterschiedliche **Kernelarchitekturen**)
    - Monolithische Kerne, Minimale Kerne, Hybride Kerne

Übungsblatt 1 wiederholt die für die Lernziele relevanten Inhalte dieses Foliensatzes

Einordnung der Betriebssysteme in die Informatik (1/2)

Wo würden Sie die Betriebssysteme einordnen?

Einordnung der Betriebssysteme in die Informatik (2/2)



Nebenfach

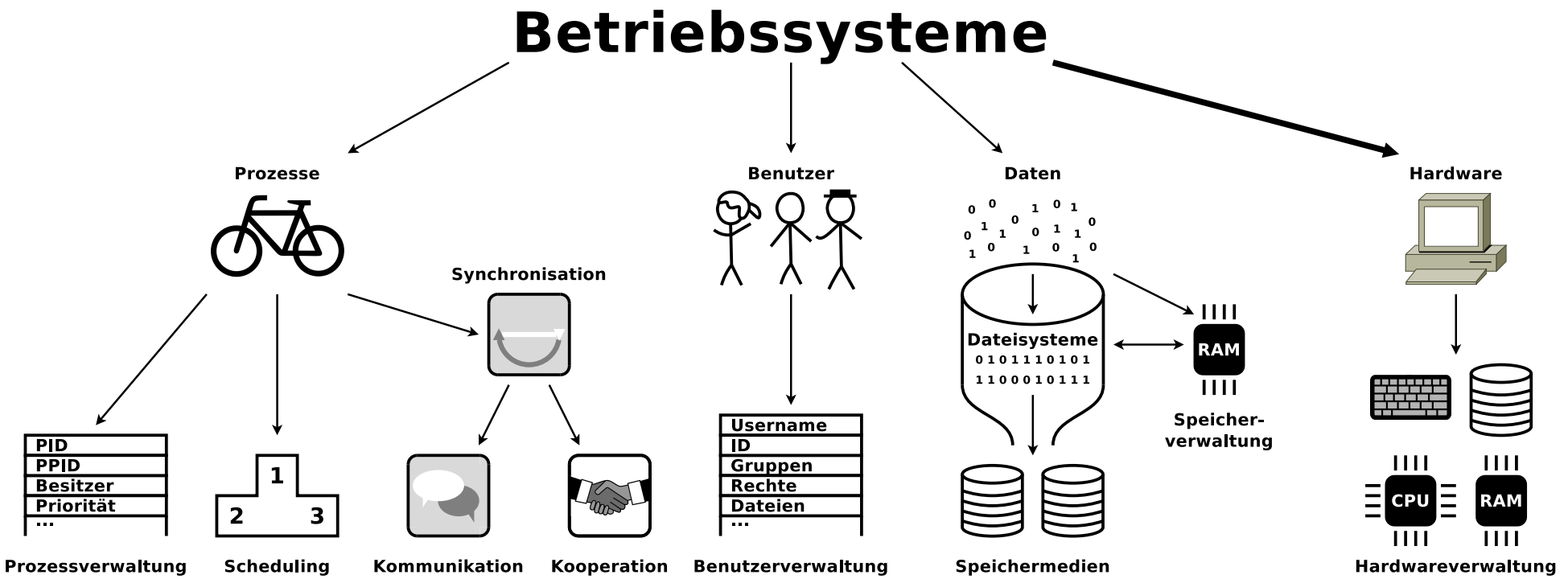
E-Technik

BWL/VWL

Medizin

Betriebssysteme gehören zur praktischen Informatik und technischen Informatik

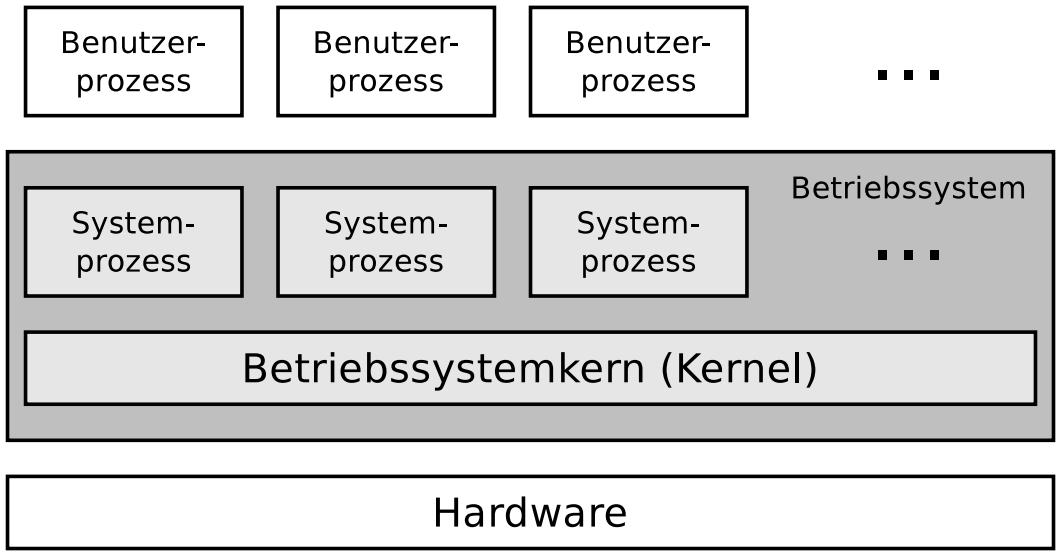
Kernfunktionalitäten von Betriebssystemen



Am Ende des Semesters...

- kennen Sie **Kernfunktionalitäten** von Betriebssystemen
- verstehen Sie die **Arbeitsweise** der Kernfunktionalitäten
- haben Sie grundlegende Kenntnisse im Umgang mit **Linux**
- haben Sie grundlegende Kenntnisse in **Shell-Programmierung**

Prinzipieller Aufbau eines Betriebssystems



- Benutzerprozesse arbeiten die Aufträge der Benutzer ab
- Systemprozesse erbringen Dienstleistungen des Betriebssystems
- Alle nicht als Systemprozesse realisierten Komponenten enthält der Betriebssystemkern ($\Longrightarrow$ Kernel)

Betriebssysteme sind ein Teil der Systemsoftware

Systemsoftware steuert den Betrieb eines Rechners, stellt eine Verbindung zur Hardware her und steuert die Verwendung und Zuteilung der Hardwareressourcen

Generationen von Computern und Betriebssystemen

Fragen, die die folgenden Folien klären sollen...

- Was für Betriebssysteme gibt es?
- Seit wann gibt es Betriebssysteme?
- Wie hat die Entwicklung der Hardware die Entwicklung der Betriebssysteme beeinflusst?

Generation	**Zeitraum**	**Technologischer Fortschritt**
0	bis 1940	(Elektro-)mechanische Rechenmaschinen $\Longrightarrow$ keine Software!
1	1940 – 1955	Elektronenröhren, Relais, Klinkenfelder
2	1955 – 1965	Transistoren, Stapelverarbeitung
3	1965 – 1980	Integrierte Schaltungen, Dialogbetrieb
4	1980 – 2000	Hoch-integrierte Schaltungen, Mikroprozessoren, PCs/Workstations
5	2000 bis ?	Verteilte Systeme, *Das Netz ist der Computer*. Virtualisierung

Aus der Zeitschrift *Populäre Mechanik* (1949)

„Computer der Zukunft werden nicht mehr als 1,5 Tonnen wiegen.“

1. Generation (1940 bis 1955)

- Die 1.Generation Computersysteme entstand während des 2.Weltkriegs
  $\Longrightarrow$ Konrad Zuse, John von Neumann
- Anforderungen an einen universellen Computer:
  - Gespeichertes Programm
  - Bedingte Sprünge
  - Trennung von Speicher und Prozessor
- Rechner waren Maschinen mit teilweise $>$ 10.000 Röhren oder Relais, die langsam und fehleranfällig arbeiteten
- **Betriebssysteme und Programmiersprachen waren unbekannt**
- Programme wurden über Steckfelder gesteckt
  - Der Benutzer/Programmierer startet **ein** Programm, dass direkt auf die Hardware zugreift

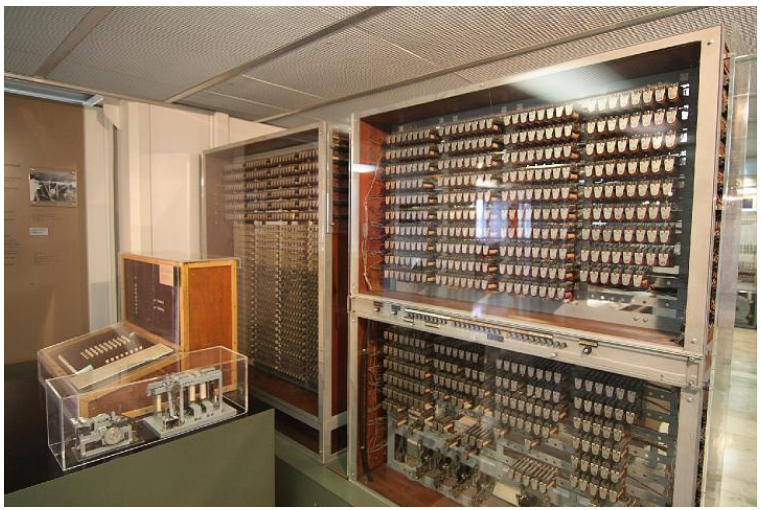
Bekannte Vertreter der 1. Generation

Bildquelle: Eigenes Werk (12.12.2008)

Maschine	Entwicklung	Speicher/CPU getrennt	bedingte Sprünge	Program-mierung	interne Kodierung	Zahlen-darstellung	Technologie
Z1 / Z3	1936-1941	ja	nein	SW	binär	Gleitkomma	Mechanisch (Relais)
ABC	1938-1942	ja	nein	HW	binär	Festkomma	Elektronisch
Harvard Mark 1	1939-1944	nein	nein	SW	dezimal	Festkomma	Elektronisch
ENIAC	1943-1945	nein	teilweise	HW	dezimal	Festkomma	Elektronisch
Manchester	1946-1948	ja	ja	SW	binär	Festkomma	Elektronisch
EDSAC	1946-1948	ja	ja	SW	binär	Festkomma	Elektronisch

Computer, die intern nach dem Dezimalsystem arbeiten?

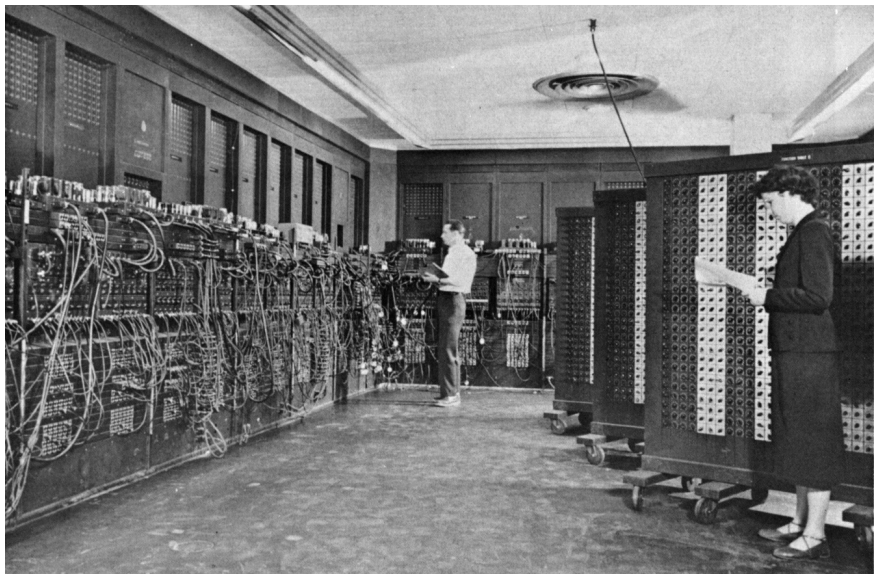
Detaillierte Beschreibung des Aufbaus: `http://computer-modell-katalog.de/eniac.htm`

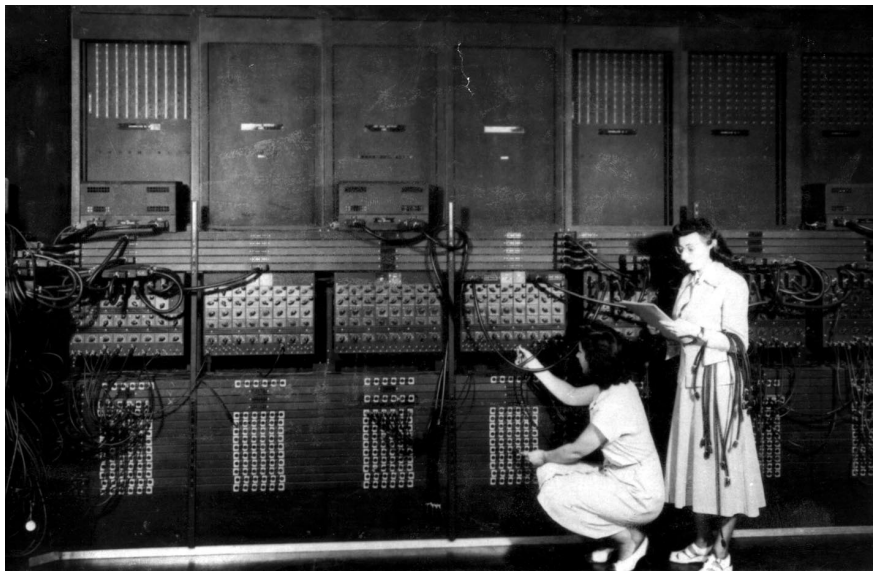


- Bild: Zuse Z3 (1941)
- Erster programmierbarer Digitalrechner der Welt (basiert auf Relaistechnik)
- Erstmals Verwendung des Dualsystems

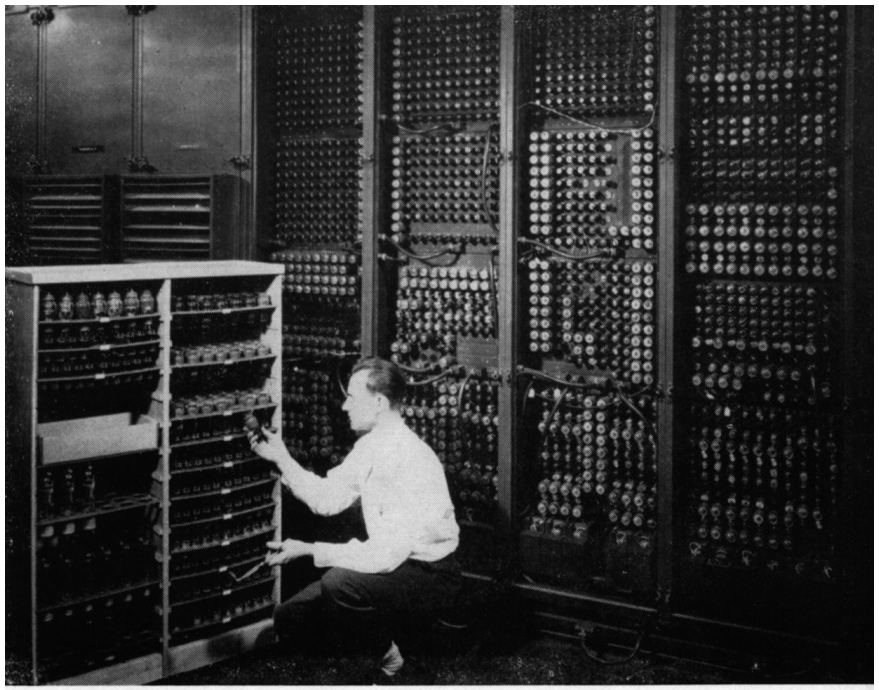
1. Generation: ENIAC (1944)

Bildquelle: US Army (Public Domain)





- Electronic Numerical Integrator and Computer (ENIAC)
- Erster elektronischer Universalrechner (mit Elektronenröhren)



Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

2. Generation (1955 bis 1965)

Bildquelle: Flickr (born1945, CC-BY-2.0)

- Anfang der 1950er Jahre: Lochkarten lösen die Steckfelder ab
- Mitte der 1950er Jahre: Einführung der Transistoren
  $\Longrightarrow$ Rechnersysteme werden zuverlässiger

- FORTRAN- oder COBOL-Programme wurden...
  - vom Programmierer auf Formblätter aufgeschrieben,
  - vom Eingeber bzw. Codierer in Lochkarten gestanzt
  - und dem Operator übergeben
- Der Operator...
  - koordiniert die Reihenfolge der Programme (Jobs)
  - bestückt den Rechner mit den Lochkarten
  - lädt den Compiler vom Magnetband
  - übergibt das Rechenergebnis als Ausdruck
    $\Longrightarrow$ Ineffiziente Arbeitsweise
- Später wurden aus Effizienzgründen die Programme gesammelt, auf Magnetbänder eingelesen und dann im Maschinenraum verarbeitet

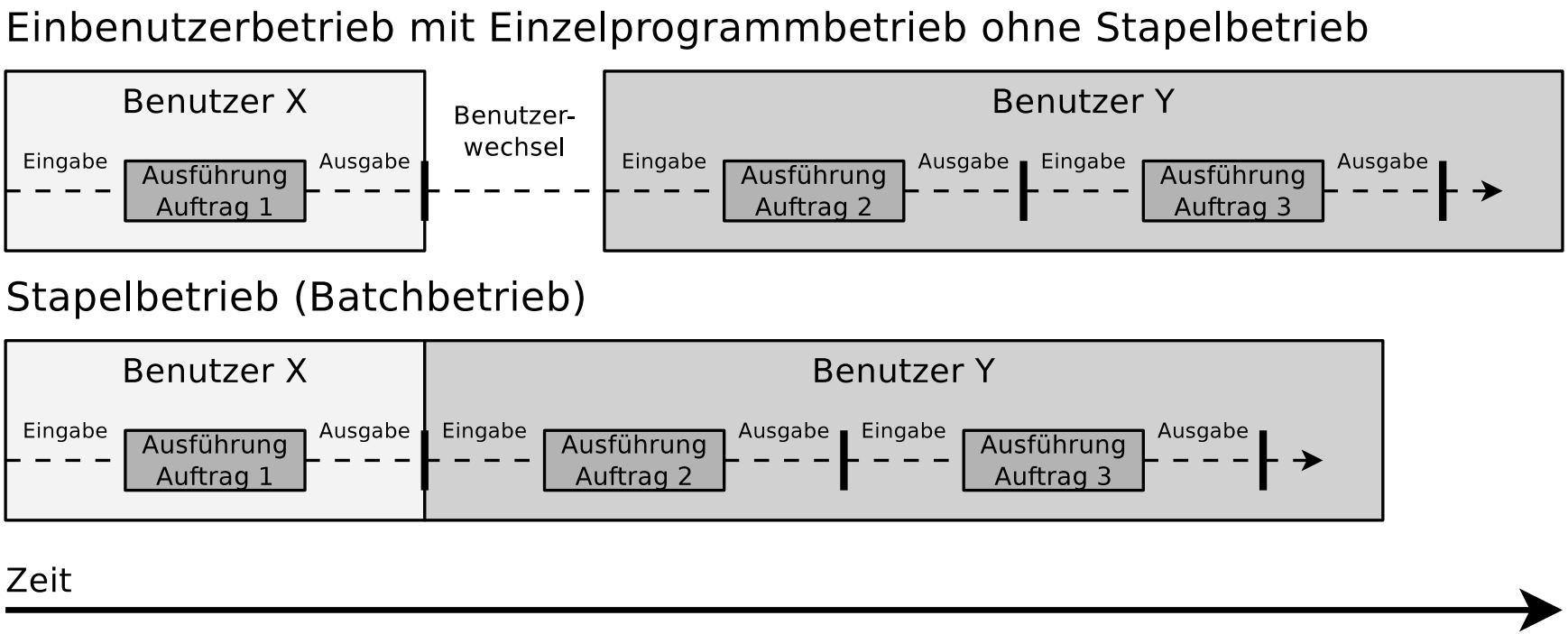
2. Generation: Stapelbetrieb bzw. Batchbetrieb (1/4)

- Frühe Betriebssysteme waren **Stapelverarbeitungs-Betriebssysteme**
- Ziel: **Maximierung der Prozessorausnutzung**

- Jedes Programm muss (mit allen Eingabedaten!) **vollständig vorliegen**, bevor die Abarbeitung beginnen kann
- Stapelbetrieb eignet sich gut zur Ausführung von **Routineaufgaben**
- Auch heutige Systeme ermöglichen die automatische Bearbeitung von Programmabfolgen (z.B. Batch-Dateien, Shell-Skripte, usw.)

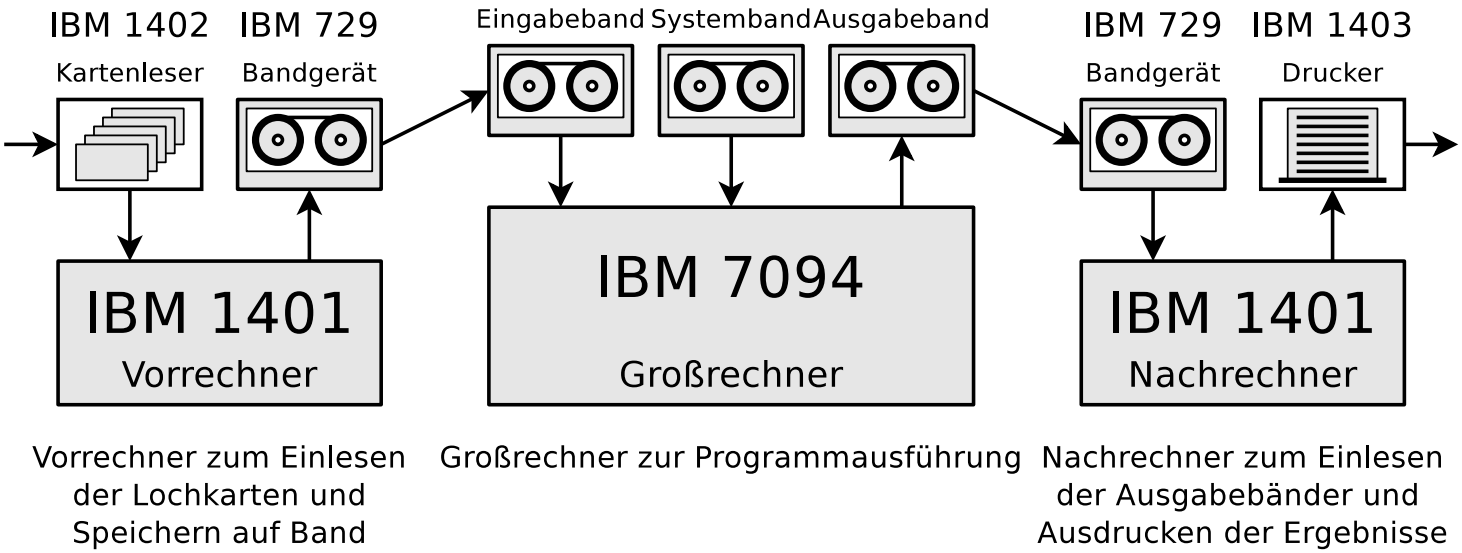
Bildquelle: IBM (Das Bild zeigt eine IBM 7090 von 1959)
`http://www.computer-history.info/Page4.dir/pages/IBM.7090.dir/images/ibm.7090.jpg`

2. Generation: Stapelbetrieb bzw. Batchbetrieb (2/4)



- Stapelbetrieb $\Longrightarrow$ **Beschleunigung durch Automatisierung**
- Nachteil: Der Hauptprozessor wird noch nicht optimal ausgenutzt
  - Während der Ein-/Ausgabe liegt der Prozessor brach

2. Generation: Stapelbetrieb bzw. Batchbetrieb (3/4)

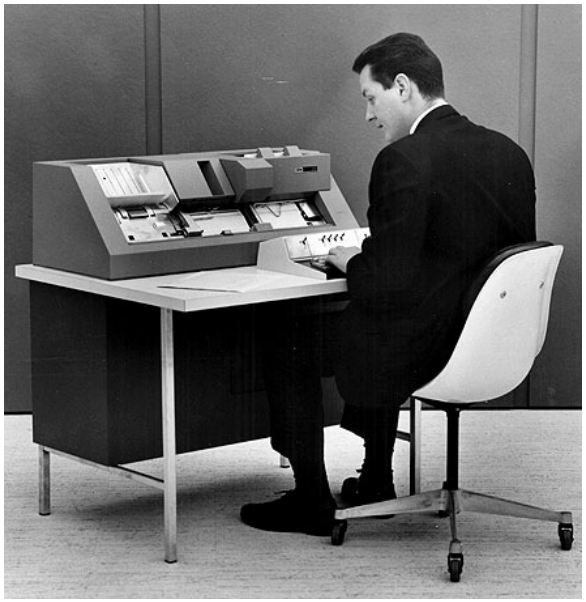


- Vor-/Nachrechner befreien den Großrechner von langsamer I/O-Arbeit
  - Von Band kann schneller eingelesen werden, als von Lochkarten und auf Band kann schneller Ausgegeben werden als auf Papier
- **Spooling** ist die Entlastung des Hauptprozessors durch zusätzliche Hardware für Ein-/Ausgabeoperationen
  - Ein-/Ausgabe geschieht nebenläufig zur Bearbeitung anderer Aufträge

Moderne Computer haben neben der CPU spezielle, DMA-fähige (*Direct Memory Access*) Ein-/Ausgabeprozessoren

Diese schreiben Daten direkt in den Hauptspeicher und holen von dort Ergebnisse

2. Generation: Stapelbetrieb bzw. Batchbetrieb (4/4)



Bildquelle: IBM Archives
`https://onfoss.com/a-timeline-of-computer-interface-technology/`

- Spooling ist heute noch aktuell
  - z.B. Spoolingprozesse zum Drucken
- Üblicherweise ist Stapelverarbeitung (Batchbetrieb) **interaktionslos**
  - Nach dem Start eines Programms wird dieses bis zum Ende oder Auftreten eines Fehlers ohne Interaktion mit dem Benutzer abgearbeitet

Stapelbetrieb ist heute nicht obsolet!

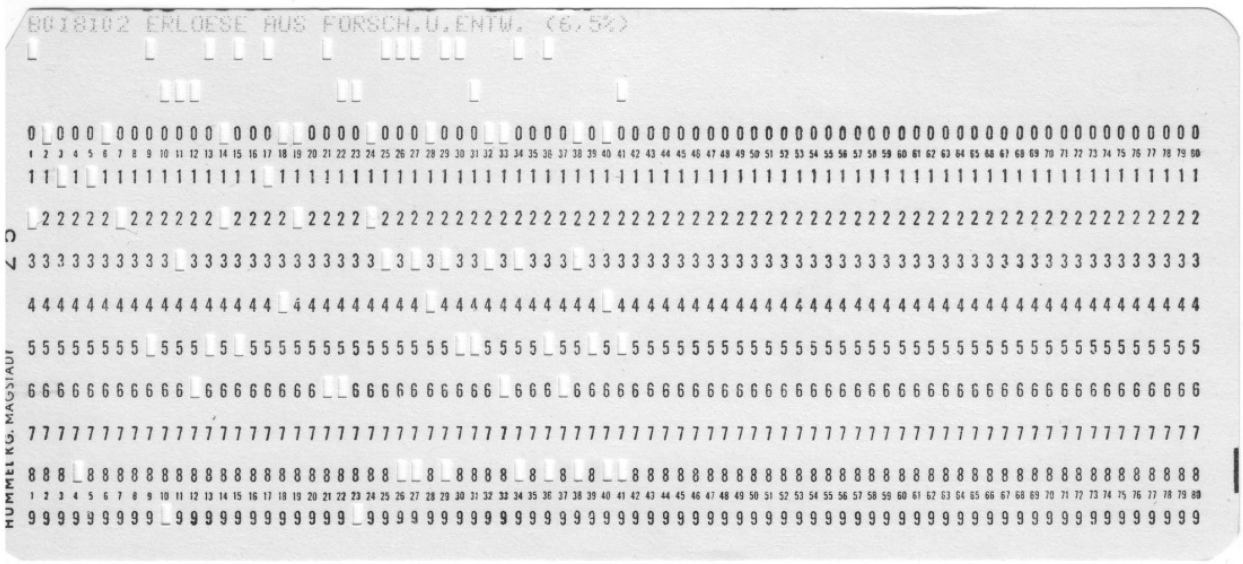
Rechenintensive Programme in verteilten Systemen sind meist interaktionslose Batchprogramme
$\Longrightarrow$ Distributed Computing und sog. Number Crunching

- Stapelverarbeitungs-Betriebssysteme der zweiten Generation bieten nur **Einzelprogrammbetrieb** = **Singletasking** ($\Longrightarrow$ Folie 26)
  - Das Betriebssyst. erlaubt nur die Ausführung eines Programms auf einmal
  - Der Start eines zweiten Programms erfordert die Beendigung des Ersten

Einige Betriebssysteme der 2. Generation

Atlas Supervisor, GM-NAA I/O, UMES, SHARE, IBSYS

2. Generation: Lochkarten



- Jede Lochkarte stellt üblicherweise eine Zeile Programmtext mit 80 Zeichen oder entsprechend viele binäre Daten dar
  - Das die Zeilenlänge von E-Mails und Textdateien heute noch typischerweise 80 Zeichen beträgt, geht auf die Lochkarte zurück
- 12 Lochpositionen für die Kodierung jedes Zeichens
  - Ziffern kodiert man mit einem einzelnen Loch in der entsprechenden Zeile
  - Buchstaben und Sonderzeichen kodiert man, indem mehrere Löcher in die Spalte gestanzt werden

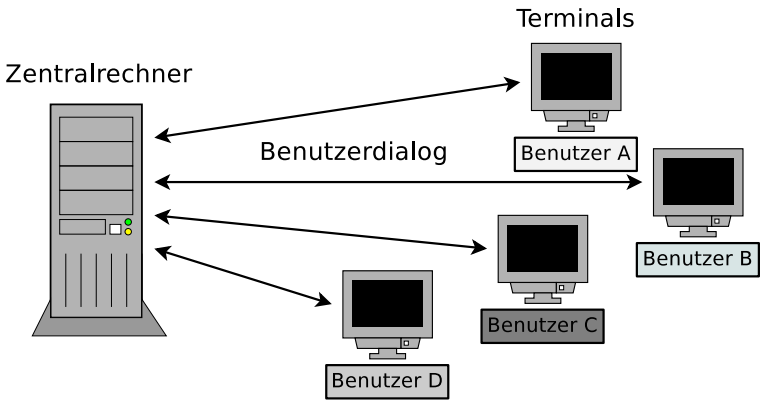
3. Generation (1960 bis 1980)

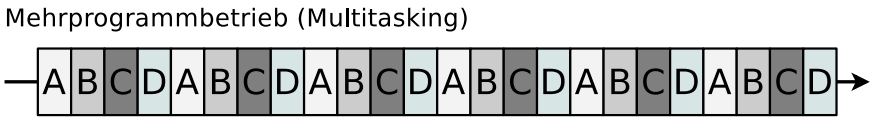
- Frühe 1960er Jahre: Integrierte Schaltungen sind verfügbar
  $\Longrightarrow$ Leistungsfähigere, kleinere und billigere Computer
- 1960er Jahre:
  - Weiterentwicklung der Stapelverarbeitungssysteme um mehrere Jobs gleichzeitig abzuarbeitende $\Longrightarrow$ **Multitasking**
  - Erste einfache **Speicherverwaltung** (*Fixed Partitions*) $\Longrightarrow$ Foliensatz 2
- 1970er Jahre: **Dialogbetrieb** (*Time Sharing*) bzw. Zeitteilbetrieb
  - Eine Zentraleinheit, mehrere Terminals (Dialogstationen)
  - Jeder Benutzer erhält beim Anmelden einen Benutzerprozess
- Ende der 1970er Jahre: Entwicklung des Mikroprozessors
  $\Longrightarrow$ Entwicklung des Heimcomputers / Personal Computers (PC)
  - 1977: Apple II. Erster Heimcomputer
  - 1981: IBM PC. Meist verkaufte Rechnerarchitektur (Intel 80x86)

Einige Betriebssysteme der 3. Generation

BESYS, CTSS, OS/360, CP/CMS, Multics, Unics (später Unix), Version 6/7 Unix, DEC CP/M, Cray OS, DEC VMS

3. Generation: Dialogbetrieb – Time Sharing (1/2)





- **Mehrere Benutzer** arbeiten an einem Computer **gleichzeitig und konkurrierend**, indem sie sich die verfügbare Rechenzeit des Hauptprozessors teilen
  - Ziel: Faire Verteilung der Rechenzeit
- Verteilung der Rechenzeit durch **Zeitscheiben** (*Time Slices*)
  - Die Verteilung kann nach unterschiedlichen Strategien erfolgen
- **Mehrere Benutzer** können **gleichzeitig** über Terminals am Computer **interaktiv** arbeiten $\Longrightarrow$ **Mehrbenutzerbetrieb** ($\Longrightarrow$ Folie 28)
- Die Programme der einzelnen Benutzer sind unabhängig voneinander
- Die quasi-parallele Programm- bzw. Prozessausführung heißt **Mehrprogrammbetrieb** oder **Multitasking** ($\Longrightarrow$ Folie 26)
  - Ziel: Minimierung der Antwortzeit

3. Generation: Dialogbetrieb – Time Sharing (2/2)

- Durch Dialogbetrieb wurden neue Konzepte nötig:
  - **Speicherschutz**: Der Arbeitsspeicher wird aufgeteilt und laufende Programme voneinander getrennt
    - So können Programmierfehler oder der Absturz eines Programms nicht die Stabilität anderer Programme und des Gesamtsystems beeinträchtigen

    $\Longrightarrow$ Foliensatz 2
  - **Dateisysteme**, die quasi-gleichzeitige Dateizugriffe erlauben

    $\Longrightarrow$ Foliensatz 3
  - **Swapping** (*Umlagerung*): Prozess des Ein- und Auslagerns von Daten in den/vom Arbeitsspeicher vom/in den Auslagerungsspeicher (Festplatten/SSDs)

    $\Longrightarrow$ Foliensatz 4
  - **Scheduling**: Automatische Erstellung eines Ablaufplanes (*schedule*), der Benutzern bzw. Prozessen zeitlich begrenzte Ressourcen zuteilt

    $\Longrightarrow$ Foliensatz 5

4. Generation (1980 bis 2000)

- Hochintegrierte Schaltkreise und exponentiell wachsende Integrationsdichte der elektronischen Komponenten
  - Prozessoren werden immer leistungsfähiger und preiswerter
  - Speicherbausteine haben eine immer höhere Kapazität
- Hohe Rechenleistung kann an jedem Arbeitsplatz installiert werden
  - Workstations setzten sich im professionellen Umfeld durch
  - Immer größerer Erfolg von Heimcomputern und Personal Computern
    - Hauptaufgabe der Betriebssysteme: Bereitstellung **intuitiver Benutzeroberflächen** für Benutzer, die von der zu Grunde liegenden Hardware nichts wissen wollen

Einige Betriebssysteme der 4. Generation

QDOS, Xenix, MS-DOS, PC-DOS, QNX, GNU-Projekt, SunOS, MacOS, AmigaOS, Atari TOS, Windows, IBM AIX, GEOS, SGI IRIX, MINIX, OS/2, NeXTSTEP, SCO UNIX, Linux, BeOS, Haiku, Google Fuchsia

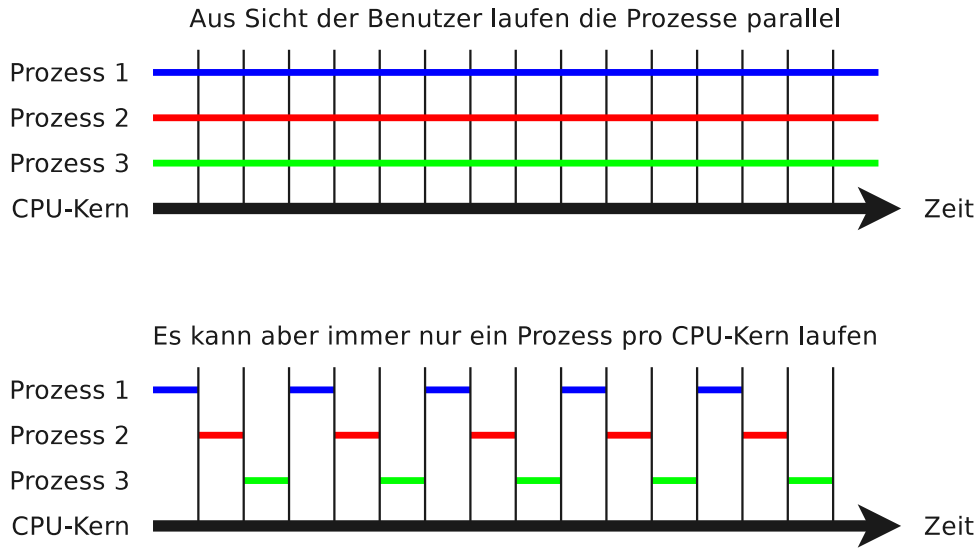
- Etablierung von Computernetzen mit offenen Standards
  - Ethernet, Token Ring, WLAN ($\Longrightarrow$ Foliensätze 7 und 8)

5. Generation (2000 bis ????)

- Einige Schlagworte aus der 5. Generation:
  - *Das Netz ist der Computer*
  - Verteilte Systeme $\Longrightarrow$ Cluster-, Cloud-, Grid-, P2P-Computing
  - Multicore-Prozessoren und parallelisierte Anwendungen
  - Virtualisierung $\Longrightarrow$ **VMware, XEN, KVM, Docker...**
  - Freie Software (OpenSource) $\Longrightarrow$ **Linux (Android), BSD,...**
  - Kommunikation überall $\Longrightarrow$ mobile Systeme
  - Neue Arbeitsformen $\Longrightarrow$ e-Science, e-Learning, e-Business,...
  - Dienste und Services $\Longrightarrow$ Web Services (REST, SOAP, JSON)
  - Ressourcen nach Bedarf mieten bzw. anfordern $\Longrightarrow$ On Demand
  - Personal Computing vs. Parental Computing (z.B. iOS)
  - Künstliche Intelligenz = Artificial Intelligence (AI)
- Schlagworte für später:
  - Quantencomputer (evtl. 6. oder 7. Generation)

Einzelprogrammbetrieb und Mehrprogrammbetrieb

- **Einzelprogrammbetrieb** (Singletasking)
  - Zu jedem Zeitpunkt läuft nur ein einziges Programm
  - Mehrere gestartete Programme werden **nacheinander** ausgeführt
- **Mehrprogrammbetrieb** (Multitasking)
  - Mehrere Programme können **gleichzeitig** (bei mehreren CPUs/Kernen) oder **zeitlich verschachtelt (quasi-parallel)** ausgeführt werden



Task, Prozess, Aufgabe, Auftrag,...

Der Begriff **Task** ist gleichzusetzen mit **Prozess** oder aus Anwendersicht **Aufgabe** bzw. **Auftrag**

Warum Mehrprogrammbetrieb (Multitasking)?

Wir wissen...

- Bei **Mehrprogrammbetrieb** laufen mehrere Prozesse nebenläufig
- Die Prozesse werden in kurzen Abständen, abwechselnd aktiviert
  $\Longrightarrow$ Dadurch entsteht der **Eindruck der Gleichzeitigkeit**
- Nachteil: Das Umschalten von einem Prozess zu anderen, erzeugt **Verwaltungsaufwand (Overhead)**

- Prozesse müssen häufig auf äußere Ereignisse warten
  - Gründe sind z.B. Benutzereingaben, Eingabe/Ausgabe-Operationen von Peripheriegeräten, Warten auf eine Nachricht eines anderen Programms
  - Durch Mehrprogrammbetrieb können Prozesse, die auf ankommende E-Mails, erfolgreiche Datenbankoperationen, geschriebene Daten auf der Festplatte oder ähnliches warten, in den Hintergrund geschickt werden
    - **Andere Prozesse kommen so früher zum Einsatz**
- **Der Overhead**, der bei der quasiparallelen Abarbeitung von Programmen durch die Programmwechsel entsteht, **ist im Vergleich zum Geschwindigkeitszuwachs zu vernachlässigen**

Einzelbenutzerbetrieb und Mehrbenutzerbetrieb

- **Einzelbenutzerbetrieb** (Single-User)
  - Der Computer steht immer nur einem einzigen Benutzer zur Verfügung
- **Mehrbenutzerbetrieb** (Multi-User)
  - Mehrere Benutzer können gleichzeitig mit dem Computer arbeiten
    - Die Benutzer teilen sich die Systemressourcen (möglichst gerecht)
    - Benutzer müssen (u.a. durch Passwörter) identifiziert werden
    - Zugriffe auf Daten/Prozesse anderer Benutzer werden verhindert

	Single-User	**Multi-User**
Singletasking	MS-DOS, Palm OS	—
Multitasking	OS/2, Windows 3x/95/98, BeOS, MacOS 8x/9x, AmigaOS, Risc OS	Linux/UNIX, MacOS X, Server-Versionen der Windows NT-Familie

- Die Desktop/Workstation-Versionen von Windows NT/XP/Vista/7/8/10/11 sind **halbe Multi-User-Betriebssysteme**
  - Verschiedene Benutzer können nur nacheinander am System arbeiten, aber die Daten und Prozesse der Benutzer sind voreinander geschützt

8/16/32/64 Bit-Betriebssysteme

- Die Bit-Zahl gibt die **Länge der Speicheradressen** an, mit denen das Betriebssystem intern arbeitet
  - Ein Betriebssystem kann nur so viele Speichereinheiten ansprechen, wie der Adressraum zulässt
- **8 Bit-Betriebssysteme** können 2^8 Speichereinheiten adressieren
  - z.B. GEOS, Atari DOS, Contiki
- **16 Bit-Betriebssysteme** können 2^{16} Speichereinheiten adressieren
  - z.B. MS-DOS, Windows 3.x, OS/2 1.x

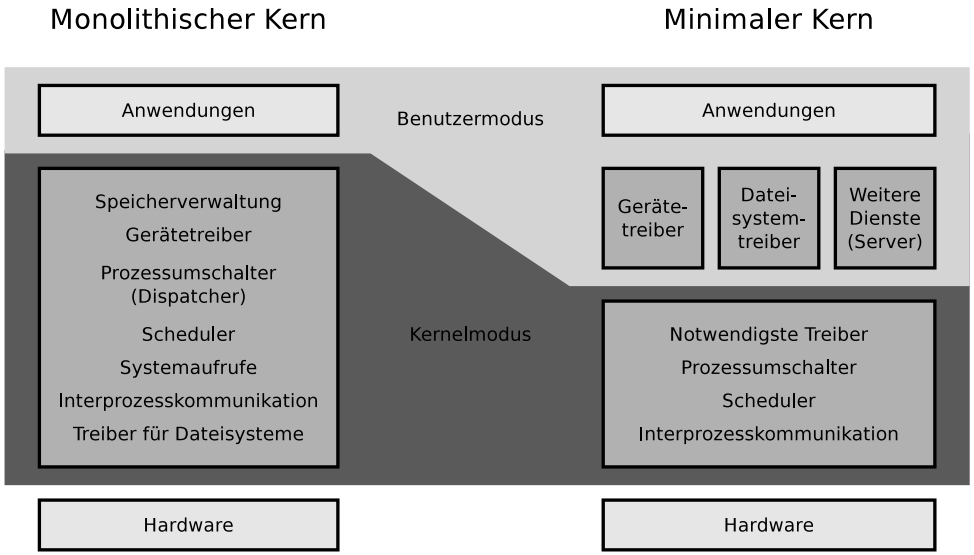
Bill Gates (1989)

„We will never make a 32-bit operating system.“

- **32 Bit-Betriebssysteme** können 2^{32} Speichereinheiten adressieren
  - z.B. Windows 95/98/NT/Vista/7/8/10, OS/2 2/3/4, eComStation, Linux, BeOS, MacOS X (bis einschließlich 10.7)
- **64 Bit-Betriebssysteme** können 2^{64} Speichereinheiten adressieren
  - z.B. Linux (64 Bit), Windows 7/8/10 (64 Bit), MacOS X (64 Bit)

Betriebssystemaufbau (Kernelarchitekturen)

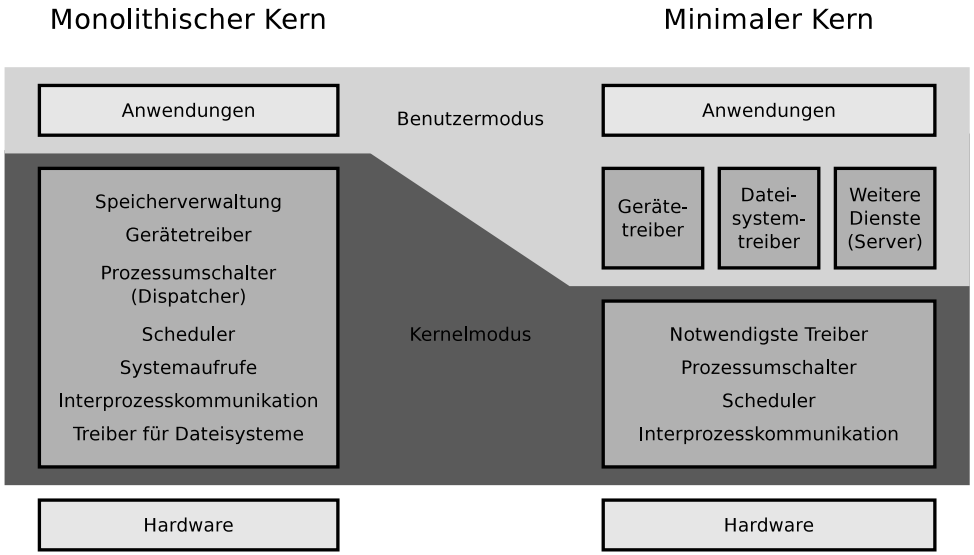
- Der **Kernel** enthält die grundlegenden Funktionen des Betriebssystems und ist die Schnittstelle zur Hardware



- Unterschiedliche Kernelarchitekturen existieren
  - Sie unterscheiden sich darin, welche Funktionen **im Kern** enthalten sind und welche sich **außerhalb des Kerns** als Dienste (Server) befinden
- Funktionen im Kern, haben vollen Hardwarezugriff (**Kernelmodus**)
- Funktionen außerhalb des Kerns können nur auf ihren virtuellen Speicher zugreifen (**Benutzermodus**)
  $\Longrightarrow$ Foliensatz 2

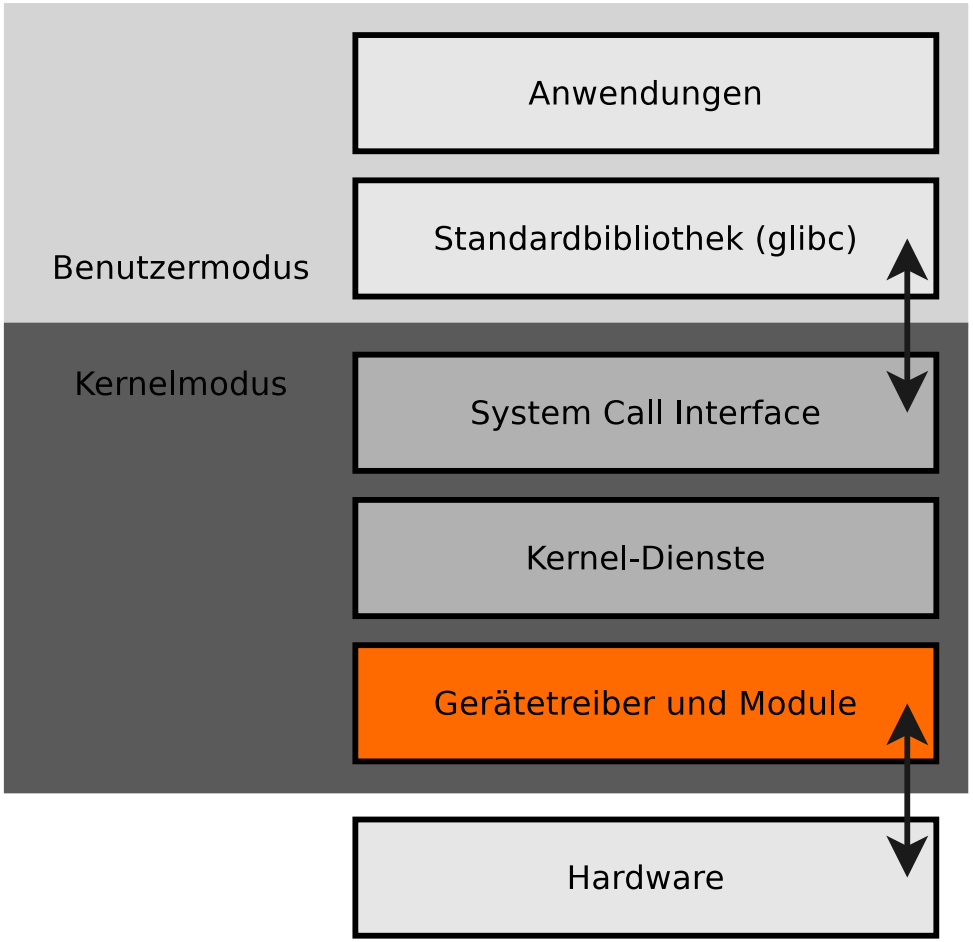
Monolithische Kerne (1/2)

- Enthalten Funktionen zur...
  - Speicherverwaltung
  - Prozessverwaltung
  - Prozesskommunikation
  - Hardwareverwaltung
  - Dateisysteme



- Vorteile:
  - Weniger Kontextwechsel als Mikrokernel $\Longrightarrow$ höhere Geschwindigkeit
  - Gewachsene Stabilität
    - Mikrokernel sind in der Regel nicht stabiler als monolithische Kerne
- Nachteile:
  - Abgestürzte Komponenten des Kerns können nicht separat neu gestartet werden und das gesamte System nach sich ziehen
  - Hoher Entwicklungsaufwand für Erweiterungen am Kern, da dieser bei jedem Kompilieren komplett neu übersetzt werden muss

Monolithische Kerne (2/2)



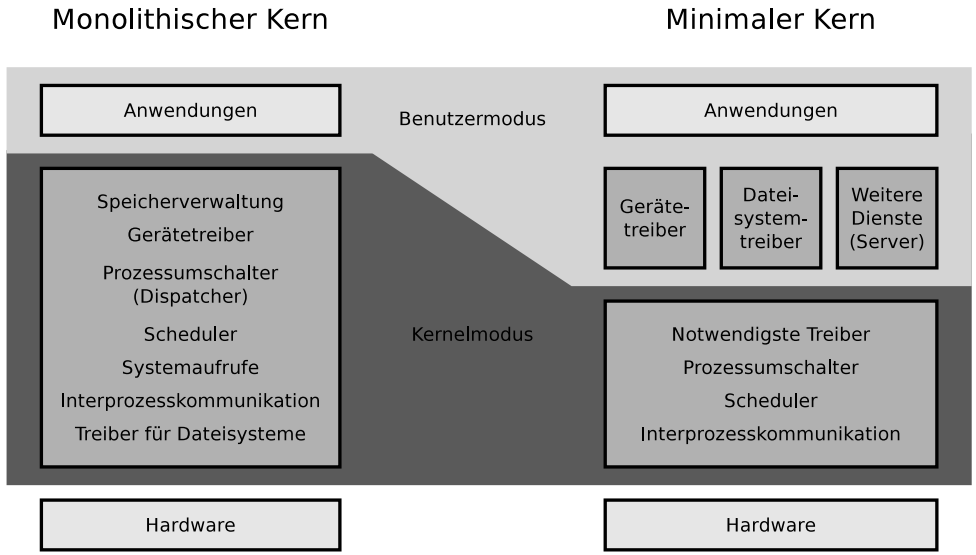
- Linux ist das populärste moderne Betriebssystem mit einem monolithischem Kern
- Hardware- und Dateisystem-Treiber im **Linux-Kernel** können in Module auslagert werden
  - Die Module laufen im *Kernelmodus* und nicht im *Benutzermodus*
    - Darum ist der Linux-Kernel ein monolithischer Kernel

Beispiele für Betriebssysteme mit monolithischem Kern

Linux, BSD, MS-DOS, FreeDOS, Windows 95/98/ME, MacOS (bis 8.6), OS/2

Minimale Kerne (1/2)

- Im Kern sind primär...
  - Funktionen zur Speicher- und Prozessverwaltung
  - Funktionen zur Synchronisation und Interprozesskommunikation
  - Notwendigste Treiber (z.B. zum Systemstart)



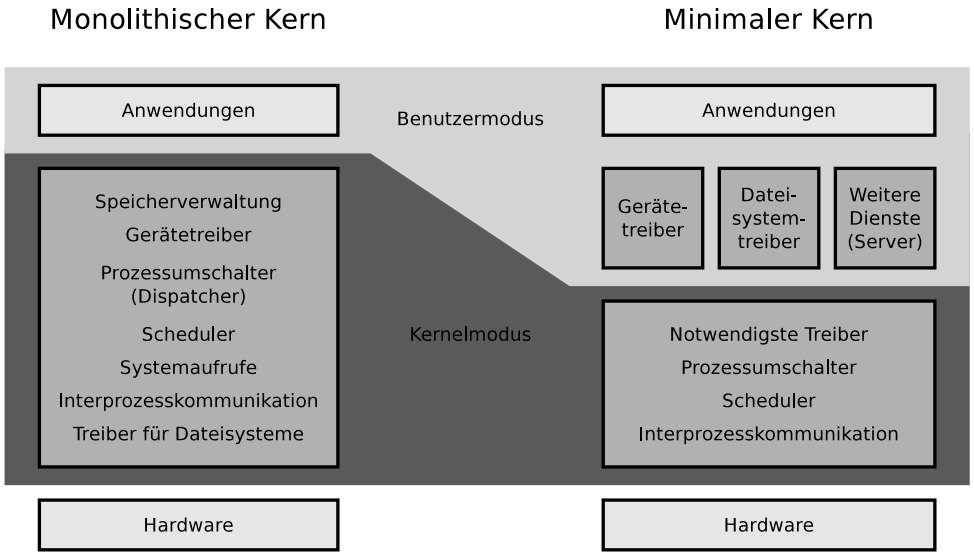
- Gerätetreiber, Dateisysteme und Dienste (Server) sind außerhalb des Kerns und laufen wie die Anwendungsprogramme im Benutzermodus

Beispiele für Betriebssysteme mit Mikrokernel

AmigaOS, MorphOS, Tru64, QNX Neutrino, Symbian, GNU HURD (siehe Folie 37)

Minimale Kerne (2/2)

- Vorteile:
  - Einfache Austauschbarkeit der Komponenten
  - Theoretisch beste Stabilität und Sicherheit
    - Grund: Es laufen weniger Funktionen im Kernelmodus



- Nachteile:
  - Langsamer wegen der größeren Zahl von Kontextwechseln
  - Entwicklung eines neuen (Mikro-)kernels ist eine komplexe Aufgabe

Der Anfang der 1990er Jahre prognostizierte Erfolg der Mikrokernelsysteme blieb aus
$\Longrightarrow$ Diskussion von Linus Torvalds vs. Andrew S. Tanenbaum (1992) $\Longrightarrow$ siehe Folie 36

Hybride Kerne / Hybridkernel / Makrokernel

- Kompromiss zwischen monolithischen Kernen und minimalen Kernen
  - Enthalten aus Geschwindigkeitsgründen Komponenten, die bei minimalen Kernen außerhalb des Kerns liegen
- Es ist nicht festgelegt, welche Komponenten bei Systemen mit hybriden Kernen zusätzlich in den Kernel einkompiliert sind
- Die Vor- und Nachteile von hybriden Kernen zeigt Windows NT 4
  - Das Grafiksystem ist bei Windows NT 4 im Kernel enthalten
    - Vorteil: Steigerung der Performance
    - Nachteil: Fehlerhafte Grafiktreiber führen zu häufigen Abstürzen

Quelle: **MS Windows NT Kernel-mode User and GDI White Paper**. `https://technet.microsoft.com/library/cc750820.aspx`

- Vorteile:
  - Bessere Geschwindigkeit als minimale Kerne da weniger Kontextwechsel
  - Höhere Stabilität (theoretisch!) als monolithische Kerne

Beispiele für Betriebssysteme mit hybriden Kernen

Windows seit NT 3.1, ReactOS, MacOS X, BeOS, ZETA, Haiku, Plan 9, DragonFly BSD

Linus Torvalds vs. Andrew Tanenbaum (1992)

Bildquelle: unbekannt

- 26. August 1991: Linus Torvalds kündigt das Projekt Linux in der Newsgroup `comp.os.minix` an
  - 17. September 1991: Erste interne Version (0.01)
  - 5. Oktober 1991: Erste offizielle Version (0.02)
- 29. Januar 1992: Andrew S. Tanenbaum schreibt in der Newsgroup `comp.os.minix`: „**LINUX is obsolete**“
  - Linux hat einen monolithischen Kernel $\Longrightarrow$ Rückschritt
  - Linux ist nicht portabel, weil auf 80386er optimiert und diese Architektur wird demnächst von RISC-Prozessoren abgelöst (Irrtum!)

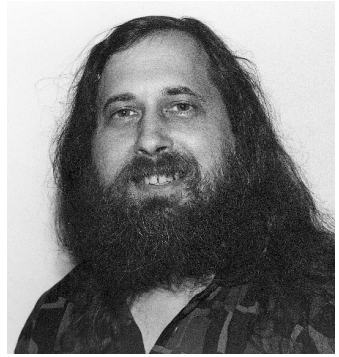
Es folgte eine mehrere Tage dauernde, zum Teilemotional geführte Diskussion über die Vor- und Nachteile von monolithischen Kernen, minimalen Kernen, Portabilität und freie Software

A. Tanenbaum (30. Januar 1992): „*I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design :-)*“. Quelle: `http://www.oreilly.com/openbook/opensources/book/appa.html`

Die Zukunft kann nicht vorhergesagt werden

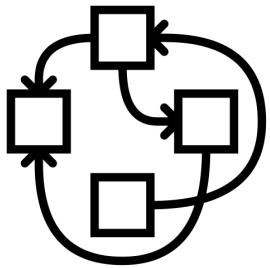
Ein trauriges Kernel-Beispiel – HURD

- 1984: Richard Stallman gründet das GNU-Projekt
- Ziel: Entwicklung eines freien UNIX-Betriebssystems
  $\Longrightarrow$ **GNU HURD**
- GNU HURD besteht aus:
  - GNU Mach, der Mikrokernel
  - Dateisysteme, Protokolle, Server (Dienste), die im Benutzermodus laufen
  - GNU Software, z.B. Editoren (GNU Emacs), Debugger (GNU Compiler Collection), Shell (Bash),...
- GNU HURD ist *so weit* fertig
  - Die GNU Software ist seit Anfang der 1990er Jahre weitgehend fertig
  - Nicht alle Server sind fertig implementiert
- Eine Komponente fehlt noch: Der Mikrokernel



Bildquelle:
`stallman.org`

Wikipedia
(CC-BY-SA-2.0)



Wikipedia
(CC-BY-SA-3.0)

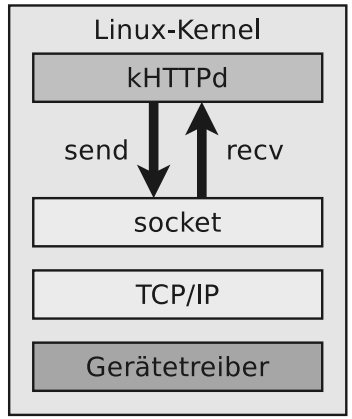
Ein extremes Kernel-Beispiel – kHTTPd

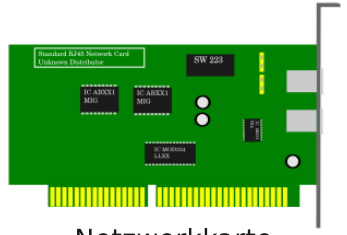
http://www.fenrus.demon.nl

- 1999: Arjan van de Ven entwickelt für Linux Kernel 2.4.x den **kernel-basierten Web-Server** kHTTPd

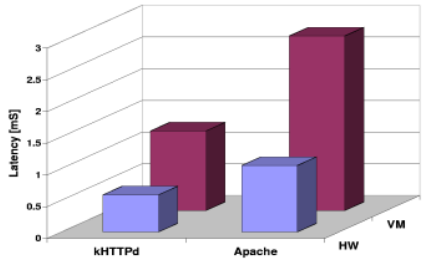
The Design of kHTTPd: https://www.linux.it/~rubini/docs/khttpd/khttpd.html
Announce: kHTTPd 0.1.0: http://static.lwn.net/1999/0610/a/khttpd.html

- Vorteil: Beschleunigte Auslieferung statischer(!) Web-Seiten
  - Weniger Moduswechsel zwischen Benutzer- und Kernelmodus
- Nachteil: Sicherheitsrisiko
  - Komplexe Software wie ein Web-Server sollten nicht im Kernelmodus laufen
  - Bugs im Web-Server könnten zu Systemabstürzen oder zur vollständigen Kontrollübernahme durch Angreifer führen
- Im Linux Kernel $\geq$ 2.6.x ist kHTTPd nicht enthalten





Netzwerkkarte



Bildquelle:
Kernel Plugins: When A VM Is Too Much. *Ivan Ganev, Greg Eisenhauer, Karsten Schwan*. 2004