Systemprogrammierung

Sommersemester 2025

Odej Kao

FG Verteilte Systeme und Betriebssysteme
Fakultät für Informatik und Elektrotechnik

1. Organisatorisches

- Team
- Kursmodalitäten
- Literatur
- Betriebssysteme im Alltag

Selbststudium

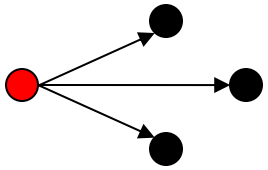
- Eine kurze Geschichte der Betriebssysteme

Das Orga-Team

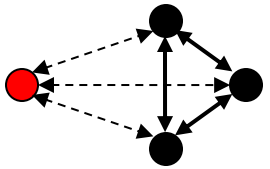
Name	E-mail	Raum	Tel.	Sprechzeiten
Dozent				
Prof. Odej Kao (DOS)	odej.kao@tu-berlin.de	EN 353	314-25154	n.V.
Sekretariat				
Jana Bechstein (DOS)	jana.bechstein@tu-berlin.de	EN 354	314-25154	Mo – Do: 12.00-16.00
Koordination				
Jonathan Rau (DOS) Justus Flerlage (DOS)	j.rau.1@tu-berlin.de Emails bitte mit „[SysProg]“ im Betreff kennzeichnen.	EN 342		n.V.

Kommunikationswege

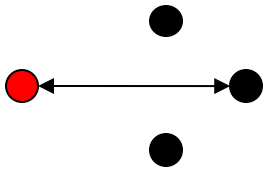
- Fachgebiets-Homepage für Lehre
  - https://www.tu.berlin/dos/studium-lehre/lehrveranstaltungen
  - Allgemeine Informationen



- ISIS: Lernplattform der Vorlesung
  - https://isis.tu-berlin.de/course/view.php?id=37659
  - Ankündigungen, Videos, Folien, Übungsblätter, Materialien, allgemeines Diskussionsforum, Gruppenforum



- E-Mail
  - Dringende Ankündigungen, Terminabsprachen
  - Individuelle Fragen und Probleme



- Persönliche Kommunikation

Vorlesungsstil: Präsenz

- Vorlesung Freitags 12-14 Uhr
  - Beispiele
  - Beantworten von Fragen

- Die PDFs der Vorlesungsfolien werden wochenweise (Freitags, nach der Vorlesung) auf ISIS hochgeladen, zusammen mit Multiple-Choice Tests (keine Bewertung, zum Selbsttest)

Vorlesungsplan

Tag	Thema	Übung
25-Apr	Organisatorisches + Prozesse 1	-
2-May	Prozesse 2 + Shell	C Basics
9-May	Scheduling	C Basics
16-May	Semaphore 1	Scheduling
23-May	Semaphore 2	Prozesssynchronisation
30-May	BM-Verwaltung	Prozesssynchronisation
6-Jun	Speicher 1	BM-Verwaltung
13-Jun	Speicher 2	Speicherverwaltung
20-Jun	Dateisysteme	Speicherverwaltung
27-Jun	Sicherheit	Dateisysteme
4-Jul	Betriebssysteme und Fallbeispiele	Wiederholung und Klausurvorbereitung
11-Jul	Virtualisierung	
18-Jul	Microservices	

Übungsbetrieb

- Beginn der Tutorien: ab dem 28.04.2025
- Inhalte
  - Ausgewählte Themen der Vorlesung werden in sechs Übungsblättern vertieft
  - Präsenztutorien für die Bearbeitung der Aufgaben
  - Zusätzlich gibt es Aufgaben zur selbstständigen Bearbeitung
  - Für Rückfragen und Lösungswege gibt es Tutorensprechstunden
- Hausaufgaben
  - Je zwei Theorie- und Programmieraufgaben
  - Einzelabgaben
- Anmeldung zu den Tutorien über Moses bis 16.04.2025

Tutorien: Termine

	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
08-10	Tutorium		Tutorium	Tutorium	Tutorium
10-12	Tutorium	Tutorium	Tutorium		Tutorium
12-14	Tutorium		Tutorium	Tutorium	Vorlesung
14-16	Tutorium		Tutorium		Tutorium
16-18	Tutorium	Tutorium	Tutorium	Tutorium	

Prüfung des Moduls Systemprogrammierung

- Einordnung in Studienplan: Pflichtveranstaltung im Bachelor Informatik und Technische Informatik
- Portfolioprüfung:
  - Hausaufgabe (50 Portfoliopunkte)
    - Note der Übung stellt 50% der Gesamtnote
    - Zwei Theorie- und Programmieraufgaben (erste Abgabe in der 5. Semesterwoche)
  - Schriftlicher Test (50 Portfoliopunkte)
    - Schriftlicher Test stellt 50% der Gesamtnote
    - Termin wird auf Isis bekanntgegeben
    - Zwei Termine, wählbar bis zum 30.06.2025

Anmeldung im MTS bis 25.05.2025, 23:59

Betriebssysteme

- Die älteste und am weitesten entwickelte Software bzgl.
  - Funktionalität
  - Effizienz
  - Ausfallsicherheit
  - Sicherheit
  - Portabilität
  - ...

- Grundlegende Mechanismen, die für viele andere Domains (Datenbanken, Grafik, Compiler, verteilte Systeme, ...) adaptiert wurden

Literatur

- Stallings, W.: Operating Systems: Internals and Design Principles, Pearson,
- Silberschatz, A. et al.: Operating System Concepts, John Wiley
- Tanenbaum, A.; Bos, H.: Modern Operating Systems, Pearson
- Zugriff auf E-Books aus dem Campusnetz der TU, ggf. per VPN.
  - Operating Systems (Campuslizenz): https://www.pearson-studium.de/drm/reader/nu/code/qz8qb0436ui6wyos6hbnas4xs2cso4ai
  - Operating Systems: Internals and Design Principles: https://ebookcentral.proquest.com/lib/tuberlin/detail.action?docID=5833157
  - Modern Operating Systems: https://ebookcentral.proquest.com/lib/tuberlin/detail.action?docID=5833738

Selbststudium

Ein schneller Durchlauf durch die Geschichte der Betriebssysteme

Geschichte der Betriebssysteme

- Entwicklung der Betriebssysteme ist eng mit der Entwicklung der Rechnerarchitekturen verbunden
- Die dargestellten Generationen von Rechnerarchitekturen sind nicht exakt und teilweise überlappend
- Erzählung ist USA- (und insbesondere MIT-)zentriert; vergleichbare Systeme entstanden auch andernorts

Erste Generation (1945-1955)

- Computerarchitekturen mit Vakuumröhren und Trommelspeicher
- Betriebssysteme und Programmiersprachen noch unbekannt (selbst Assembler wird erst im Lauf der Epoche entwickelt)
- Programmierung daher in Maschinensprache
- Folglich auch Portierung als Konzept noch unbekannt

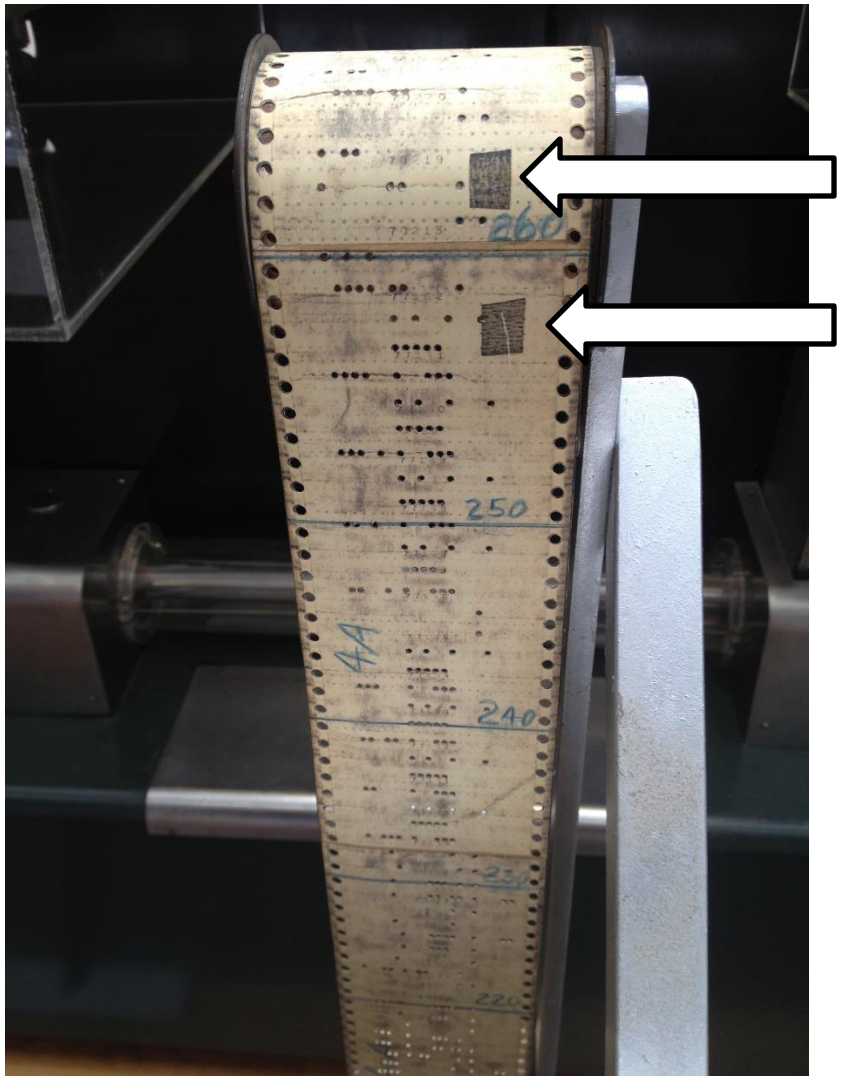
- ein (sehr positiv gezeichnetes) Bild dieser Programmierweise als Folklore: siehe „Story of Mel“

http://catb.org/jargon/html/story-of-mel.html

Zweite Generation (1955-1965)

- Transistoren führen zu erhöhter Zuverlässigkeit
- Magnetbänder als schnellerer Massenspeicher
- Programmiert wird mit Lochkarten
- Stapelverarbeitung als Hauptbetriebsart
  - Manuell: Operateur lädt neuen Job, wenn der alte abgearbeitet wurde
  - Automatisch: Kleiner, billiger Rechner kopiert Jobs auf Band; Mainframe führt Job um Job von Band aus und schreibt Resultate auf anderes Band; kleiner Rechner druckt Ausgabeband aus
- Job Control Language: Automatisierung von Teilaufgaben des Operateurs (Quellcode einlesen, Compiler laden, kompilieren, ausführen) → JCL-Interpretation ist schon eine Form von BS-Funktionalität

Randnotiz: Warum man Software „patcht“...



Lochstreifenprogramm...

© ArnoldReinhold, Wikimedia Commons #34872964

Dritte Generation (1965-1980)

- Einführung von integrierten Schaltungen
- Existenz zweier, zueinander inkompatibler Produktlinien
  - (36-bit-)wortorientierte Rechner für numerische Berechnungen in Wissenschaft und Industrie, z.B. IBM 7094
  - (6-bit-)zeichenorientierte Rechner für Sortieren und Ausdrucken von Bändern in Banken und Versicherungen, z.B. IBM 1401
  - Portabilitätsprobleme bei Erweiterung/Upgrade des Rechnerpools
- Abhilfe: IBM System/360
  - Serie von softwarekompatiblen Rechnern unterschiedlicher Leistungs-stärke mit derselben Architektur und identischem Instruktionssatz
- Widersprüchliche Anforderungen führen zu großem, komplexem und fehlerbehaftetem Betriebssystem

  → Einsatz hoher Programmiersprachen zur Implementierung von BS

  → Geburtsstunde von Software-Engineering

Randnotiz: eine IBM 1401 in Aktion

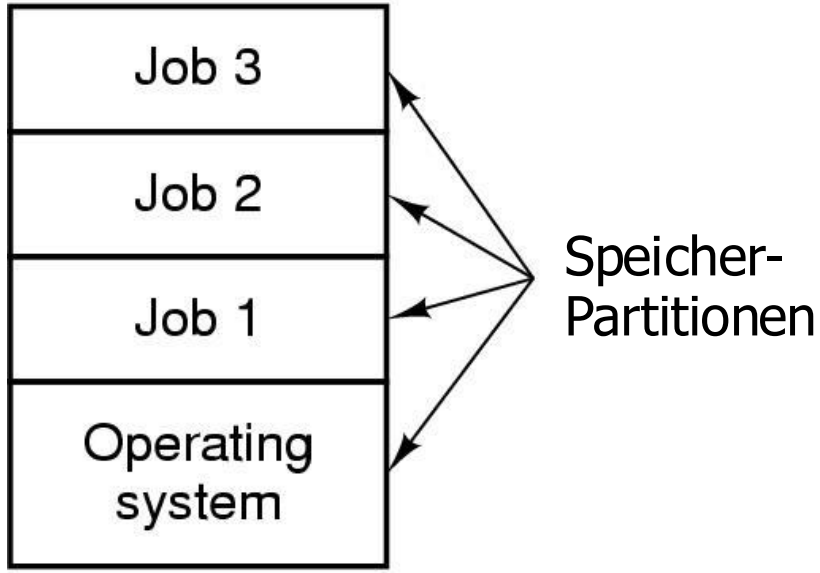
https://hackaday.com/2018/02/11/ibm-1401-runs-fortran-ii-once-more/

CTSS und der Beginn des Time-Sharings

- Forscher am MIT frustriert vom Warten auf Job-Ausgabe
- Idee des „Time-Sharings“ anstelle von Batch Processing: mehrere Nutzer können mit dem Rechner interagieren
- Kommunikation mit dem Mainframe über Terminals
- Entwicklung des Compatible Time Sharing Systems (1961/62)
  - Statt Maximierung der CPU-Auslastung Minimierung der Antwortzeit
  - Bedurfte vieler Hardware-Änderungen an der verwendeten IBM 7094
    - Intervallzeitgeber
    - zweite Speicherbank
    - zusätzliche Register für einen Speicherschutzmechanismus

CTSS und der Beginn des Time-Sharings

- Ablauflogik von CTSS:
  - Prozessor wechselt ständig zwischen Jobs
  - auf E/A wartende Jobs werden blockiert
  - Speicherbereiche der Jobs vor anderen Jobs geschützt
  - wenn kein interaktiver Job aktiv → Batch-Job „im Hintergrund" ausführen (deswegen *Compatible* TSS)



- Eingeführte Schlüsseltechnik: der Prozessor wird virtualisiert
  → der Job wird zum *Prozess*

CTSS → MULTICS

- MULTICS (*MULTIplexed Information and Computing System*)
  - System entwickelt von Bell Labs, MIT und General Electric
  - Grundidee entstammt der Stromversorgung: bei Bedarf steckt man einen Stecker in die Dose und bezieht Elektrizität
  - Analog mit Rechenleistung → MULTICS sollte ausreichend Rechenleistung für alle Einwohner von Boston bieten

- Viele Features von MULTICS heute selbstverständlich:
  - baumartiges Dateisystem, Möglichkeit, Daten zwischen Nutzern zu teilen
  - Multiprocessing
  - Speicherverwaltung auf Seitenbasis (Paging)
  - umfangreiche Bibliothek mit Hilfsfunktionen

MULTICS → UNICS

- MULTICS-Projekt sehr ambitioniert, konnte nicht gemäß Zeitplan fertiggestellt werden
- Bell Labs verließ das Projekt 1969
- GE verkaufte seine Computer-Sparte 1970
- MIT führte MULTICS zum Teilerfolg weiter

- MULTICS-Entwickler Ken Thompson suchte sich eine neue Beschäftigung und schrieb UNICS (später UNIX) für DEC PDP-7 Kleinrechner
- Bald unterstützten ganze Abteilungen von Bell Labs das Projekt und entwickelten Portierungen für diverse PDP-Geräte

Real Programmers ☺

- Dennis Ritchie (stehend) und Ken Thompson vor einem PDP-11 System
  Quelle: http://www.psych.usyd.edu.au/pdp-11/real_programmers.html

Verbreitung von UNIX

- UNIX-Portierung auf neue Maschinen ist wegen der Assemblersprache aufwendig

  → Entwicklung in einer hohen Programmiersprache notwendig

- Sprachen wie B (BCPL) reichten nicht aus, z. B. keine byteweise Adressierung
  - Ritchie entwickelt C und schreibt mit Thompson UNIX in C neu

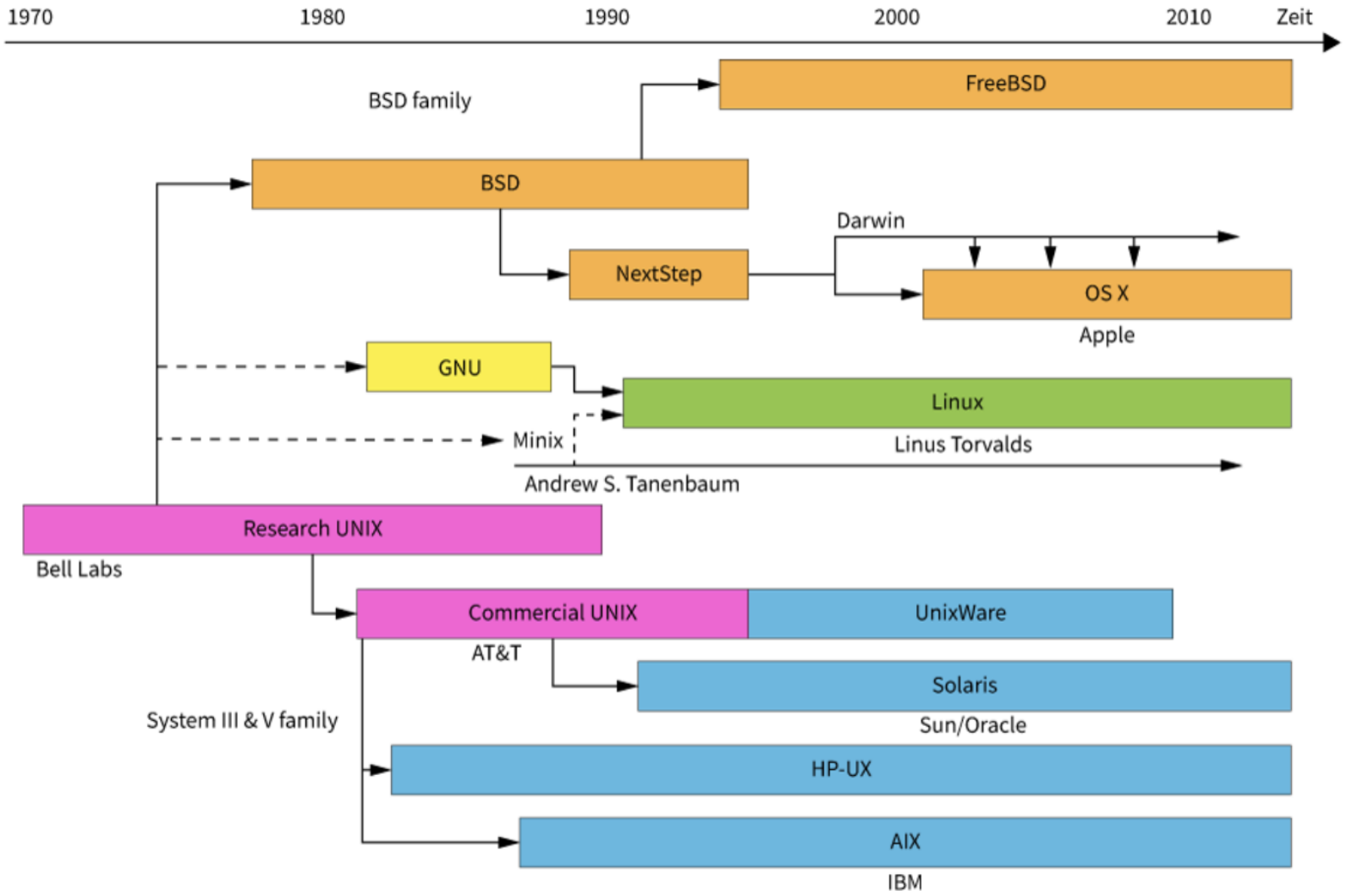
  → C als Standard für Systemprogrammierung entsteht

- AT&T durfte nicht ins Computergeschäft einsteigen

  → UNIX wurde an Universitäten inklusive Quellcodes ausgeliefert
  - Stand: 8200 Zeilen C-Code und 900 Zeilen Assembler-Code
  - Unternehmen lizenzierten den Quellcode, um eigene Versionen von UNIX zu entwickeln (z. B. XENIX vom Startup Microsoft)

- Berkeley UNIX: Berkeley Software Distribution (BSD)
  - SUN, DEC usw. bauten ihre UNIX-Versionen auf Grundlage von BSD und nicht der „offiziellen" Version UNIX V von AT&T auf

Unix und seine Derivate



Der POSIX-Standard, MINIX und Linux

- Standardisierungsprojekt POSIX (1988, mehrfach aktualisiert)
  - Systemaufrufe aus der Schnittmenge von System V und BSD, IEEE Standard 1003.1
  - Aber die UNIX-Landschaft bleibt gespalten, vgl. „Unix Wars“ und die späteren großen Gerichtsverfahren (SCO vs. IBM, …)
- Tanenbaum entwickelt 1987 MINIX als „UNIX für Ausbildungszwecke“
  - Mikrokern umfasst minimale Funktionalität, 1600 Zeilen C-Code + 800 Zeilen Assembler
- Linus Torvalds stellt 1991 Linux 0.0.1 zur Verfügung
  - Monolithischer Ansatz, zunächst auf Intel 386 beschränkt
  - POSIX-kompatibel, frei verfügbar (GNU Public License)
  - Version 1.0 erscheint 1994 mit 150.000 Zeilen Code
  - Version 2.0 erscheint 1996 mit 630.000 Zeilen Code
  - Version 3.0 erscheint 2011 mit 11.900.000 Zeilen Code

Alles beginnt mit einem ersten Schritt

> Newsgroup: comp.os.minix
> Subject: What would you like to see most in minix?
> Summary: small poll for my new operating system
> Date: 25 Aug 91 20:57:08 GMT
> Organization: University of Helsinki
>
> Hello everybody out there using minix –
>
> I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file system (due to practical reasons) among other things).
> I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)
> Linus (torvalds@kruuna.helsinki.fi)
> PS. Yes, it's free of any minix code, and it has a multi-threaded fs.
> It is NOT portable (uses 386 task switching etc.), and it probably never will
> support anything other than AT-hard disks, as that's all I have :-(.

Vierte Generation (1980-1990)

- Entwicklung von Mikrocomputern und großes Aufkommen von PCs
- Massenverbreitung von Betriebssystemen
  - CP/M (Control Program for Microcomputers)
    - Plattenbasiertes Betriebssystem für Intel 8080
    - Dominiert den Mikrocomputermarkt ca. 5 Jahre lang
  - MS-DOS (MicroSoft Disk Operating System)
    - Basiert auf DOS der Firma Seattle Computer Products
    - Wird mit IBM PC im Paket ausgeliefert → Vermarktungsstrategie entscheidender Erfolgsfaktor (CP/M wird separat verkauft)
  - Mac OS: Einführung von GUI und benutzerfreundlichen Interfaces
  - UNIX wird um eine grafische Oberfläche erweitert (X-Windows-Konzept, MIT)

Vierte Generation (2)

- Leistungsfähige Kommunikationsmedien erlauben die Einführung und Nutzung von verteilten Systemen

  → BS überwinden Rechnergrenzen: Von der Rechnerkommunikation zum Verbundsystem

- Neue Technologien, neue Herausforderungen
  - Einführung von Lightweight Processes (Threads)
  - Aufnahme von Parallelitätskonzepten in BS und Programmier-sprachen wie massiv-parallele Verarbeitung, Lastverteilung, …
  - Multimedia: Unterstützung von Bildern, Audio- und Videoströmen
  - Eingebettete Systeme: Maßgeschneiderte, effiziente, zuverlässige BS, oft mit Echtzeitanforderungen (z. B. „max. Reaktionszeit“)
  - Interoperabilität: Verteilte Systeme in heterogenen Umgebungen

Mobilgeräte (1995-...)

- Vorläufer: PDAs, elektronische Organizer mit BS
  (z. B. PalmOS, Symbian) und einfachen Anwendungen
- Erste Geräte: HP OmniGo 700LX, Nokia 9000 Communicator
  (Design noch klar PDA mit „angebautem" Telefon)
- Neue Anforderung an BS: sparsame Ressourcenverwendung (Akku-Laufzeit!) ohne spürbare Leistungseinbußen
- seit 2008: „Smartphones" (iPhone, erste Android-Geräte) lösen die „Feature Phones" ab

- iOS: entfernter Nachfahre von BSD UNIX + Mach Mikrokern
- Android: modifiziertes Linux + spezielle Systembibliotheken

Weiterentwicklung der Virtualisierung

- Virtualisierung auch der Teile des Prozessors, die das BS für Schutzmechanismen benutzt; Einführung einer neuen (höheren) Ebene Schutzmechanismen
  - neue absolute Kontrollinstanz: Hypervisor
  - Funktion und Arbeitsweise identisch zu einem BS, nur andere Ebene der Abstraktion
  - erlaubt Ausführung von mehreren BS

- Vielfältige Anwendungsfälle:
  - Entwickeln und Debuggen von Betriebssystemen
  - gleichzeitige Nutzung unterschiedlicher Software-Ökosysteme
  - Umgehen von Kompatibilitätsproblemen
  - Optimale Nutzung von Ressourcen in Rechenzentren

Virtualisierungsformen

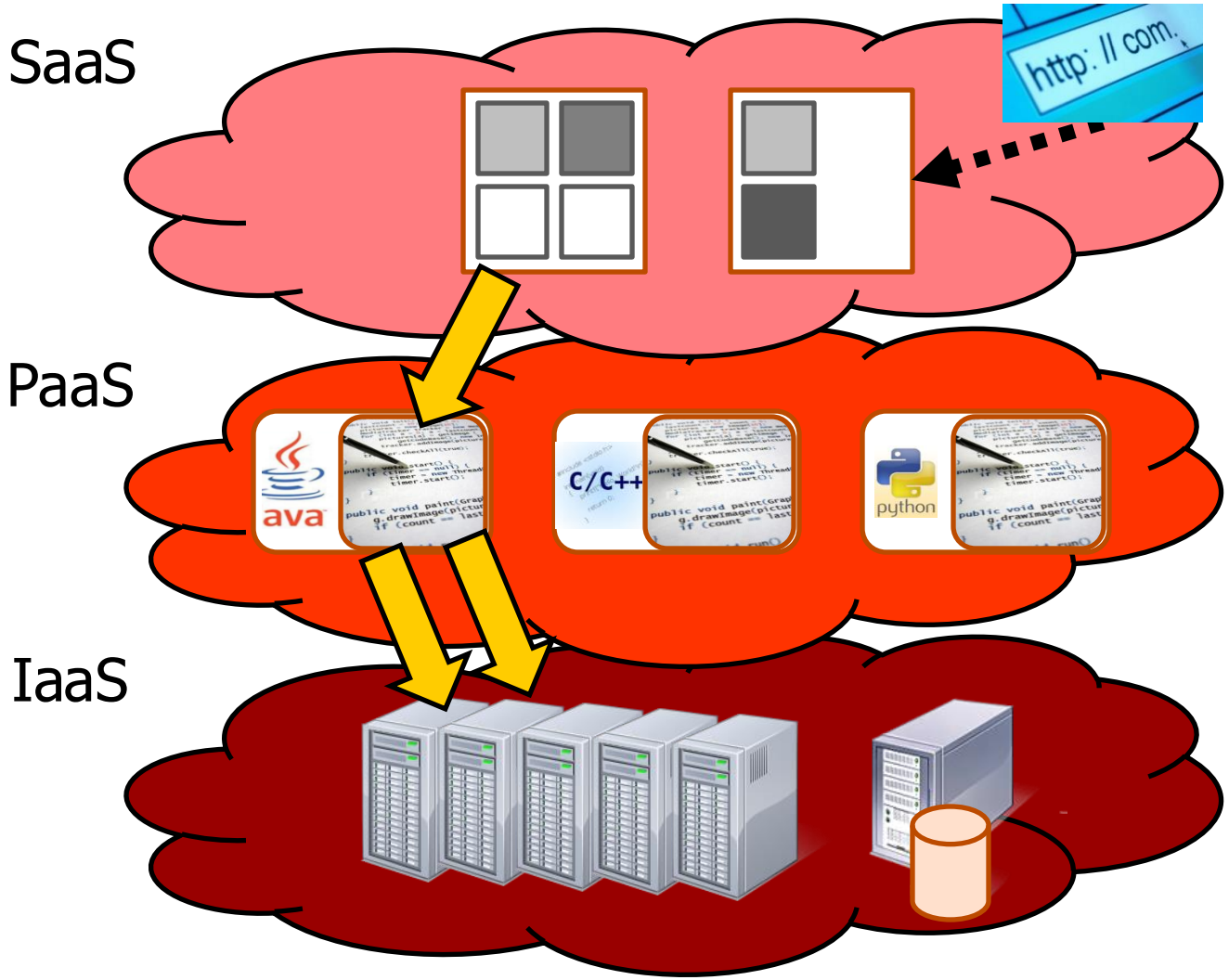
Simulation sämtlicher HW (einschließlich der CPU)	Emulation	Bochs, QEMU, PearPC
Simulation bestimmter HW-Komponenten (keine Sim. der CPU)	Full Virtualization	VMware, VirtualBox/PC
Unterstützung der Virtualisierung durch Hardwaremechanismen	Hardware Enabled Virt.	VMware Fusion, QEMU/KVM
Nur Teile der Hardware werden virtualisiert	Partial Virtualization	Hist. Systeme wie IBM M44
Keine Virtualisierung von HW, sondern API für Zugriff	Paravirtualization	Xen
(Starke) Isolation von Prozessen innerhalb eines Betriebssystems	Containerization	OpenVZ, BSD Jail, Linux cgroups
Ausführung der Anwendungen in kleiner virtueller Umgebung	Appl. Virt.	JavaVM

Cloud Computing

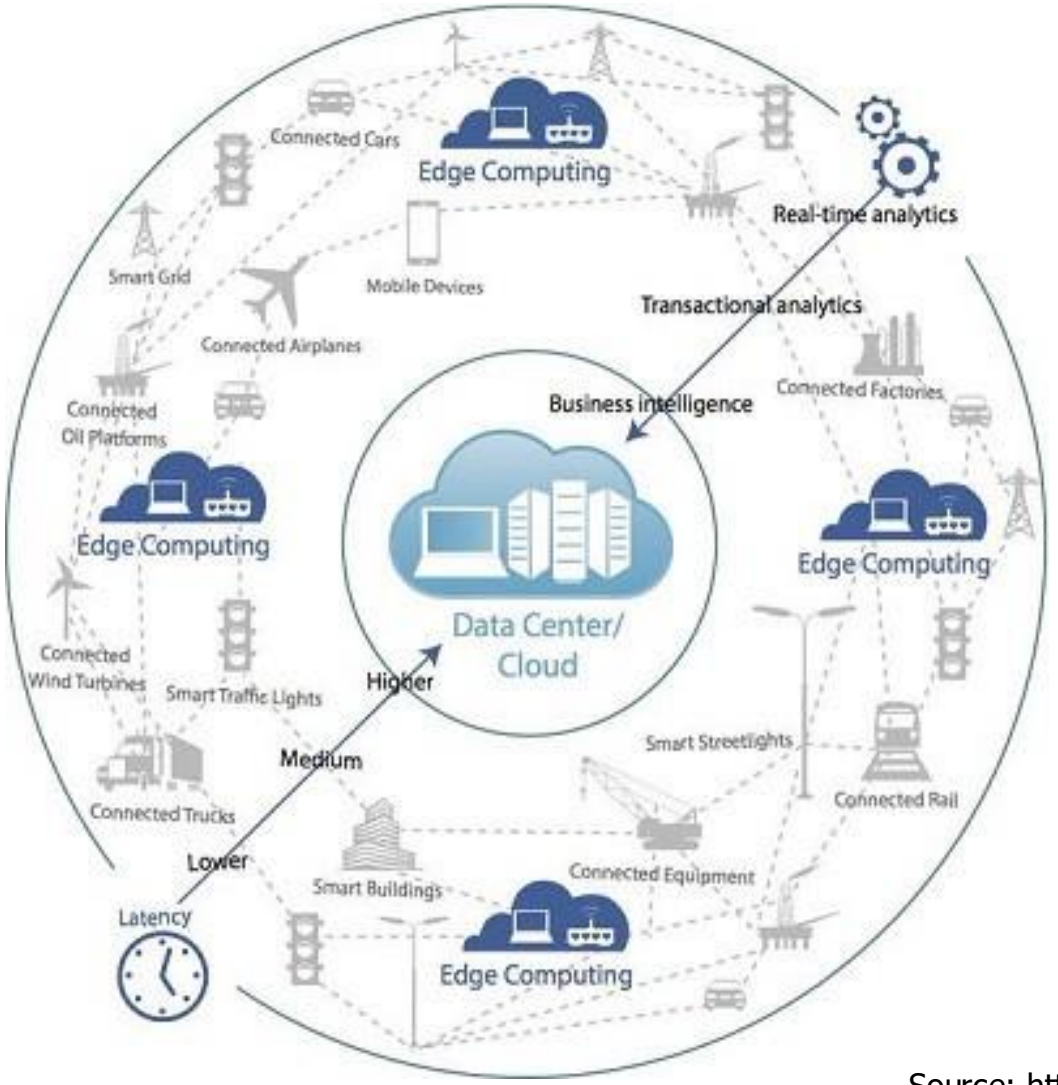
- NIST-Definition
  - Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.
    (*National Institute for Standards and Technology, NIST*)

- Zentrale Elemente in dieser Definition
  - Convenient = einfache Schnittstelle
  - On demand, rapidly provisioned = kurzfristige Bereitstellung
  - Shared Pool = Keine dedizierten Ressourcen, Multi-Tenant
  - Minimal management effort = Standard-Service des Providers

Schichten des Cloud Computing

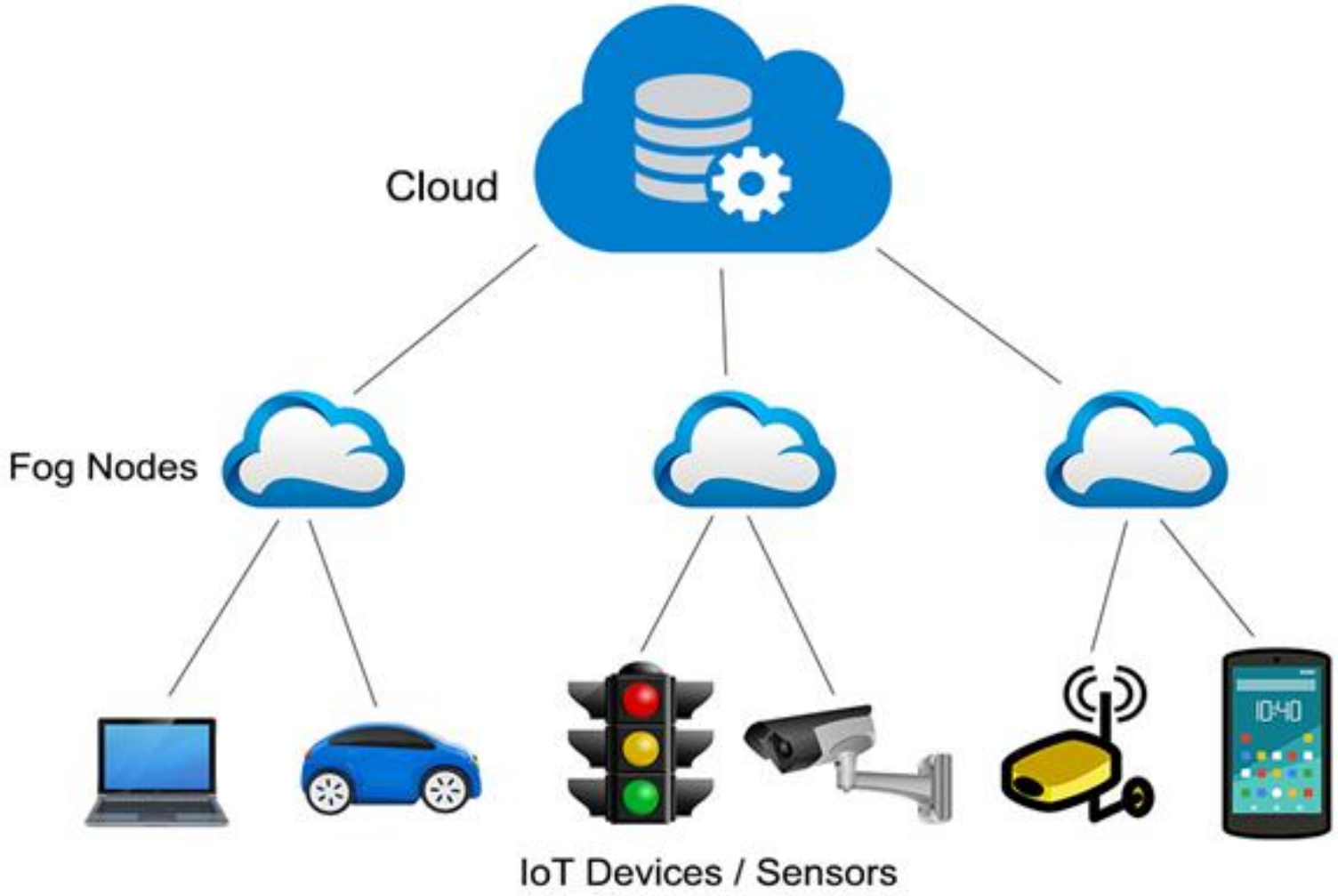


Clouds everywhere



Source: https://www.promptcloud.com/

Fog Computing



https://www.terminalworks.com/blog/post/2017/05/13/fog-computing