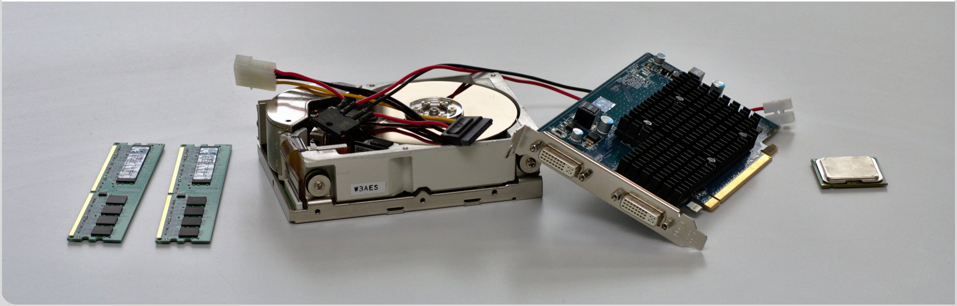
Betriebssysteme

01. Organization, History, and Hardware

Prof. Dr.-Ing. Frank Bellosa | WT 2016/2017

KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT) – OPERATING SYSTEMS GROUP



Vorlesung Betriebssysteme

- Dozent: Prof. Dr.-Ing. Frank Bellosa <bellosa@kit.edu>
- Übungsbetrieb: Dipl.-Inform. Marc Rittinghaus <rittinghaus@kit.edu>

- Vom 17.10.2016 bis 08.11.2016:
  - Montag, 17:30 - 19:00, Geb. 10.21 Benz HS
  - Dienstag, 17:30 - 19:00, Geb. 50.35 Fasanangarten HS

  Die Vorlesung **entfällt** am 1.11.2016 und 7.11.2016.
- Ab 14.11.2016:
  - Montag, 09:45 - 11:15, Geb. 10.23 Nusselt HS
  - Dienstag, 09:45 - 11:15, Geb. 20.40 Fritz-Haller HS

- Es gibt viele Möglichkeiten (fachliche) Fragen zu stellen:
  - Während der Vorlesung: zu gerade behandelten Stoff
  - Im Tutorium: zu gerade und in der Vergangenheit behandeltem Stoff
  - Im Forum (ILIAS)

Materialien

- Im ILIAS der **Vorlesung**:
  - Ankündigungen
  - Forum
  - Vorlesungsfolien
  - Übungsblätter
  - Musterlösungen
- Sie müssen nicht “pollen”
  - Email notification
  - Link ”Ankündigungen”, Menü ”Aktionen”, ”Benachrichtigungen”
  - geht auch im Forum
- Passwort: **BS16**

Tutorien I

- 18 Tutorien
- Anmeldung von Montag und bis inkl. Donnerstag via **YouSubscribe**

  `http://wiwi.link/os2016`

- YouSubscribe-Ergebnisse liegen erst am Freitag vor!
  - **In der ersten Woche in ein beliebiges Tutorium ab Mittwoch gehen!**
  - **Es gibt schon diese Woche Tutorien mit wichtigem Stoff!**

- Termine in Gebäude 50.34:

	Mi	Do	Fr
08:00 - 09:30		-118	-118
09:45 - 11:15	-109 -118	-108 -118	-118 -119 148
11:30 - 13:00	-109 -118	-108 -109	-109
14:00 - 15:30	-109 -118	-118	-109
17:30 - 19:00			

Tutorien II

- Was?
  - Diskussion der Übungsblätter
  - Vertiefung des Stoffs
  - Stellen/Beantworten von Fragen zum Stoff
- Wozu?
  - Wiederholung des VL-Stoffs der Vorwoche
  - Vorstellung weiterführender Inhalte
  - Vorbereitung auf Klausuren

Übungsblätter

- Es gibt zwei verschiedene Übungsblatttypen (wöchentlich):
  - **Übungsblätter:** Abgabe und Korrektur, unterteilt in Theorie- und Praxisteil
  - **Tutoriumsblätter:** Keine Abgabe, keine Korrektur
- Übungsblätter werden digital im ILIAS bearbeitet
  - Im Unterordner Ihres Tutoriums
  - Bearbeitungszeit in der Regel eine Woche
  - Erstes Übungsblatt nächste Woche

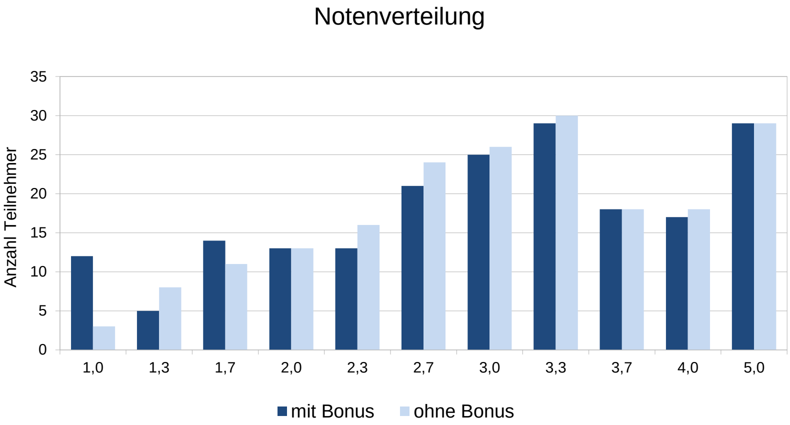
Prüfungen

- Modul ist bestanden = Übungsschein + bestandene Hauptklausur
- Modulnote = Hauptklausurnote (10.3.17 12:00 2h)
  - Elektronische Anmeldung 16.1.17 bis 5.3.17
  - Elektronische Abmeldung bis zum 9.3.17
  - Maximal eine Wiederholung der Klausur
  - Wiederholungsklausur Mitte September 2017
- Übungsschein = unbenotete Übungsscheinklausur (27.3.17 11:00 2h)
  - Elektronische Anmeldung 16.1.17 bis 19.3.17
  - Elektronische Abmeldung bis zum 26.3.17
  - Keine Wiederholungsklausur im Sommersemester
- Keine Abhängigkeiten zu anderen Prüfungsleistungen im BA Informatik
- Persönliche Abmeldung bis Klausurbeginn im Hörsaal mit Studentenausweis, KEINE Email, KEIN Telefonat, KEINE Vollmacht
- Ab Klausurbeginn Rücktritt nur mit Attest vom Tag der Prüfung unter Angabe der Diagnose. KEINE AUB!
- Keine Wiederholungsmöglichkeit einer bestandenen Leistung

Zusatztutorien

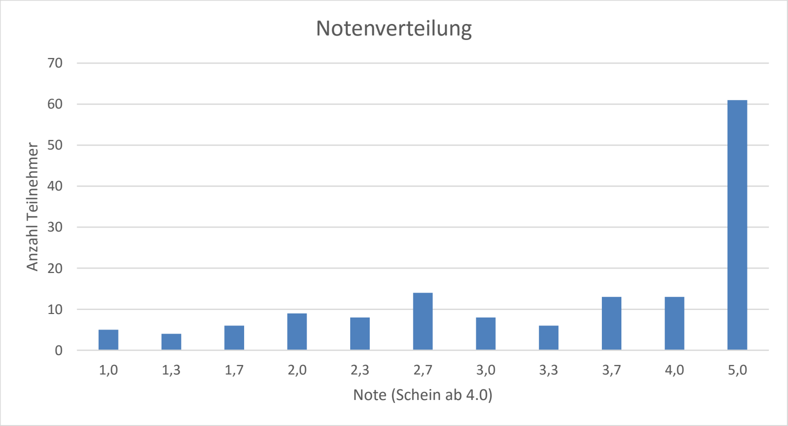
- Gelegenheit vor den Klausuren letzte Fragen zu klären.
- Zusatztutorien Hauptklausur
  - 6.3.17 13:00 2h, Raum -119
  - 7.3.17 13:00 2h, Raum -119
- Zusatztutorien Scheinklausur
  - 20.3.17 13:00 2h, Raum -119
  - 21.3.17 13:00 2h, Raum -119

Ergebnisse Hauptklausur WS 14/15



- Bestehensgrenze im WS 14/15: 25 von 60 Punkten
- Durchfallquote: 14,8%

Ergebnisse Scheinklausur WS 14/15



- Bestehensgrenze im WS14/15: 20 von 60 Punkten
- Durchfallquote: 41,5%

Inhalte

- Überblick
- Prozesse und Threads
- Prozess-Ablaufplanung
- Synchronisation
  - Multithreaded Programming
- Hauptspeicherverwaltung
- Sekundärspeicher
- Dateisysteme
- Ein- und Ausgabe

Literatur

- Remzi Arpaci-Dusseau, Andrea Arpaci-Dusseau, “Operating Systems: Three Easy Pieces”, Online Textbook
- Hillenbrand, Rittinghaus, Stöss, “Introduction to the C programming language”, Download KIT ILIAS
- Tanenbaum/Bos, “Modern Operating Systems”, 4th Edition
- Silberschatz, Galvin, Gagne, “Operating System Concepts”, 9th Edition
- Stallings, “Operating Systems – Internals and Design Principles”, 8th Edition
- Hennessy, Patterson, “Computer Architecture – A Quantitative Approach”, 5th Edition

Betriebssysteme und was dann?

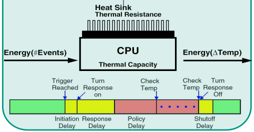
- Veranstaltungen im Bachelor Informatik

Proseminar (3 LP)

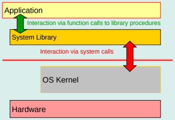
OS Internals (3 LP)

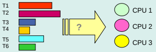
Vorlesung+Praktikum (6 LP)
(aus dem Master auf Antrag)

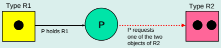
Power Management Vorlesung	Power Management Praktikum

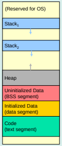


Betriebssysteme Vorlesung







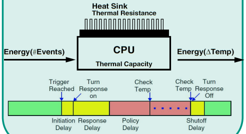


... und im Master?

- Wahlveranstaltungen im Master Informatik

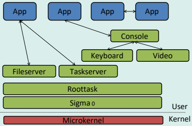
Power Management (6LP)

Power Management Vorlesung	Power Management Praktikum

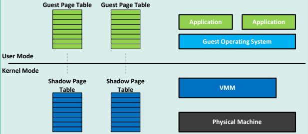


Multiserver Systems (12 LP)

Microkernel Construction Vorlesung	System Design and Implementation Vorlesung	System Design and Implementation Praktikum



Virtual Systems (3 LP)



Seminare

Systems Resource Management (3 LP)	Hot Topics in Modern Operating Systems (3 LP)	Advanced Operating Systems (Program Committee) (3+3 LP)

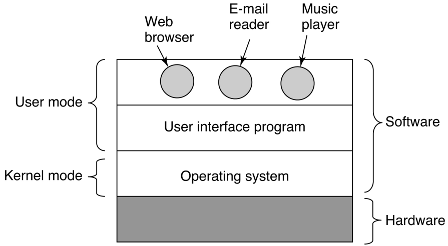
Introduction

What is an Operating System?

- If all application programmers had to understand the hardware in detail, no software would ever be written

→ Add **Operating System** Layer between applications and hardware

- Provides **abstractions** for applications
  - Manages and hides hardware details
  - Uses low-level interfaces to access hardware which are not available to applications
  - Multiplexes hardware to multiple programs ("virtualizes hardware")
  - Makes hardware use efficient for applications



- Provides **protection**
  - from users/processes using up all resources (accounting & allocation)
  - from processes writing into other processes memory

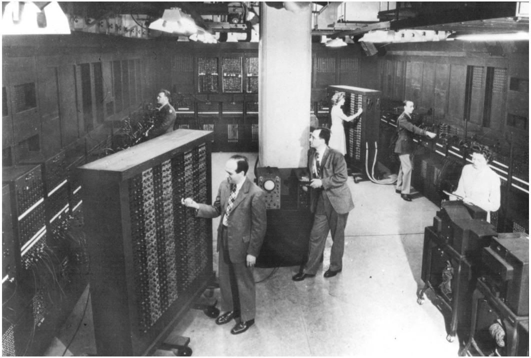
Operating System Definition

- OS is a resource manager
  - Manages and multiplexes hardware resources
  - Decides between conflicting requests for resource use
  - Strives for efficient and fair resource use
- OS is a control program
  - Controls execution of programs
  - Prevents errors and improper use of the computer
- No universally accepted definition

→ “Everything a vendor ships when you order an operating system”

History

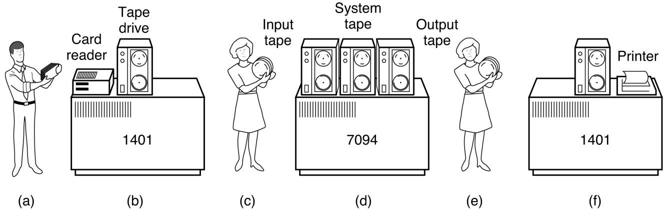
1st Generation: Vacuum Tubes and Plugboards



ENIAC 1946

- ca. 1945 – 1955
- 10,000s of vacuum tubes
- Programmed in machine language by wiring plugboards

2nd Generation: Transistors and Batch Processing



- ca. 1955 – 1965
- Programmer creates FORTRAN or assembler punchcards

(a) Operator brings punchcards to 1401
(b) 1401 reads cards and writes them to input tape
(c) Operator brings tape to 7094
(d) 7094 reads input, performs computation, stores results on output tape
(e) Operator brings output tape to 1401
(f) 1401 prints result on paper

3rd Generation: Integrated Circuits

- ca. 1965 – 1980
- Integrated Circuits (ICs) lower prices and boost performance
  - Make multiprogramming possible
- DEC PDP-7
  - 18-Bit processor
  - $ 72,000
  - 8 KiB for Unix OS
  - 16 KiB user memory for applications
- First UNIX
  - Dennis Ritchie and Ken Thompson
  - 1969 @ Bell Labs

3rd Generation: First Multiprogrammed Systems

- Multiprogramming needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Organize jobs so CPU always has one to execute
    - Job scheduler selects which job to run next
    - OS dispatches another job, when current job has to wait (e.g., for I/O)
- Timesharing/multitasking is an extreme version of multiprogramming
  - Each user can have programs executing in memory → processes
  - Switch processes so frequently, that users can interact with their jobs
    - Response time should be < 0.2 second
    - Multiple processes appear to run at the same time
  - If processes don't fit in memory, swapping moves them between background storage and memory
  - Virtual memory allows execution of processes without keeping them entirely in memory

4th Generation: Personal Computers

- ca. 1980 – present
- Large Scale Integration (LSI) circuits: $\frac{\text{thousands of transistors}}{\text{cm}^2}$
- Mass production of LSI ICs → Everybody can afford a computer now

- 1980: IBM licences MS-DOS + BASIC package from Bill Gates
  - Command Line Interface (CLI)
  - User enter command into command prompt to start programs

- ca 1988: First Graphical User Interfaces (GUIs)
  - Invented at Xerox PARC
  - First PC with GUI: Apple LISA
  - Usually mouse, keyboard, and monitor
  - Icons represent files, programs, actions, ...
  - Mouse buttons over objects in the interface cause various actions

- 1991: First Linux release

5th Generation: Mobile Computers

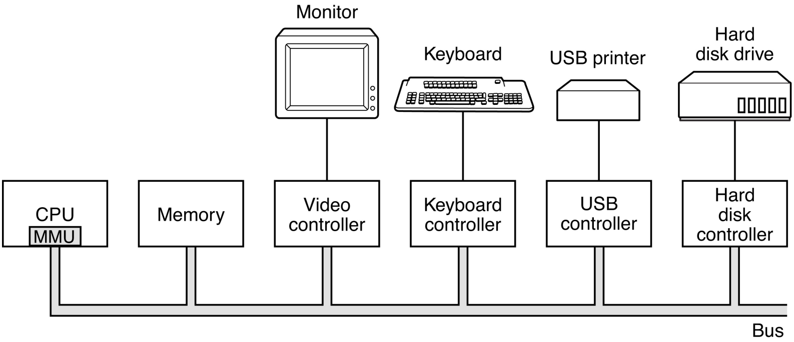
- ca. 1990 – present
- Rising IC integration density and lower power consumption make smartphones possible

- 2002: RIM introduces Blackberry running Blackberry OS
- 2007: Apple releases first iPhone with iOS Operating System
- 2008: Google releases Linux based Android Operating System

Hardware

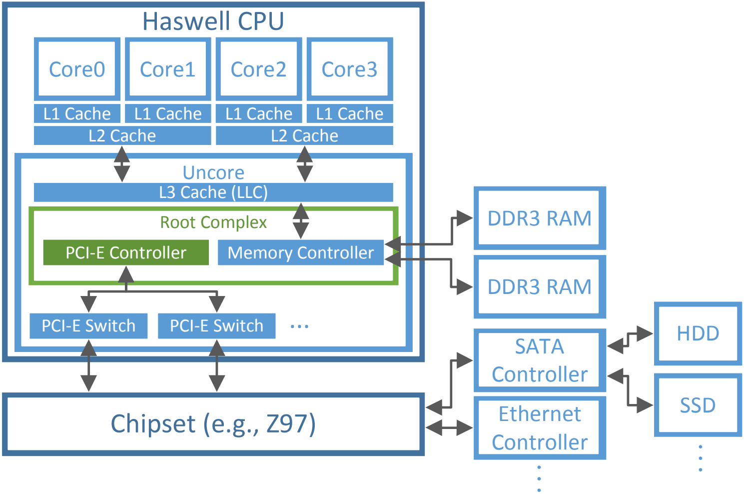
Computer System Organization

- CPU(s), devices, and memory conceptually connected to common bus
  - CPU(s) and devices compete for memory cycles and bus
  - All entities run concurrently



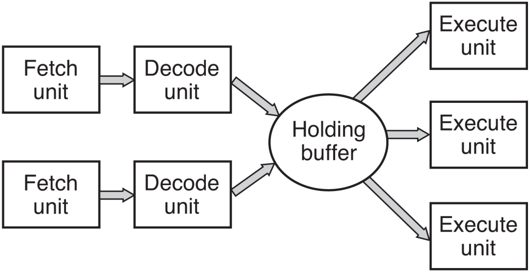
Example: Haswell

- Contemporary systems have multiple busses



Central Processing Unit (CPU) - Operation

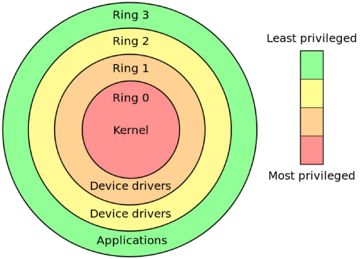
- Fetches instructions from memory and executes them
  - Instruction format and -set depends on CPU (e.g, ARM, MIPS, x86, . . .)



- CPU internal registers store data and metadata during execution
  - General purpose registers, floating point registers, . . .
  - Instruction pointer, stack pointer, . . .
  - Program Status Word (PSW)

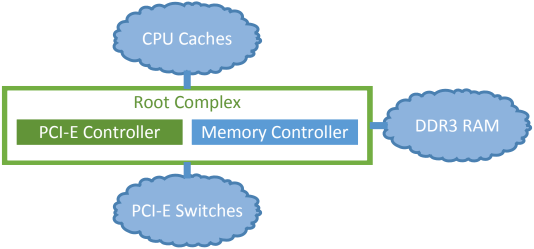
Central Processing Unit (CPU) - Modes of Execution

- User Mode (x86: “Ring 3” or CPL3)
  - Only non-privileged instructions may be executed
  - Cannot manage hardware in this mode → protection!
- Kernel Mode (x86: “Ring 0” or CPL0)
  - All instructions allowed: Can manage hardware with *privileged instructions*



Random Access Memory (RAM)

- RAM keeps currently executing instructions and data
- Today, CPUs have memory controller built-in



- Root Complex is directly connected via
  - "wire" to caches
  - pins to RAM
  - "wire" to PCI-E switches

Caching

- RAM delivers instructions/data slower than the CPU can execute them
- Memory references typically follow principle of locality
  - Spatial locality: Future references often near previous accesses (e.g., next byte in array)
  - Temporal locality: Future references often at previously accessed reference (e.g., loop counter)
- Caching helps mitigating this Memory Wall
  - Idea: Copy information in use from slower to faster storage temporarily
  - Check faster storage first before going down the Memory Hierarchy
  - If not, data is copied to cache and used from there

Typical access time		Typical capacity
1 nsec	Registers	<1 KB
2 nsec	Cache	4 MB
10 nsec	Main memory	1-8 GB
10 msec	Magnetic disk	1-4 TB

CPU Register and CPU Cache Access Latency

- CPU directly executes instructions on CPU registers
  - Register access takes ~1 CPU cycle
- Example: Intel Sandy-Bridge CPU cache hierarchy
  - L1 cache per core: ~ 4 CPU cycles
  - L2 cache per pair of cores: ~12 CPU cycles
  - L3 cache/LLC per uncore: ~28 CPU cycles (~25 GiB/s)
- DDR3-12800U RAM: 28 CPU cycles for LLC + ~50ns (~12 GiB/s)

Learn those numbers: They are often asked at job interviews!
(And come in handy when reasoning about system performance)

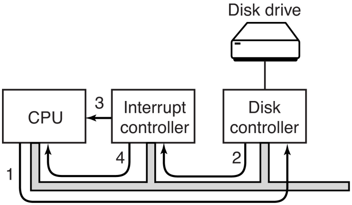
CPU Cache Organization

- Caches managed in hardware
- Caches are divided up into cache lines of typically 64 bytes each
  - Cache lines: Unit at which data is exchanged between levels of hierarchy
- Often separation of data and instructions in the faster caches (e.g., L1)
- Cache hit: Accessed data was already in the cache (e.g., L2 cache hit)
- Cache miss: Accessed data has to be fetched from lower level first
- Types of Cache misses
  - Compulsory Miss: first reference miss, data has never been accessed
  - Capacity Miss: cache is not large enough for Working Set of process
  - Conflict Miss: cache still has space, but collisions due to placement strategy

Interplay of CPU and Devices

- I/O devices and the CPU execute concurrently
- Each device controller…
  - …is in charge of a particular device
  - …has a local buffer

1. CPU issues commands and moves data to devices
2. Device controller informs the APIC that it has finished its operation
3. APIC signals the CPU
4. CPU receives device/interrupt number from APIC and executes handler

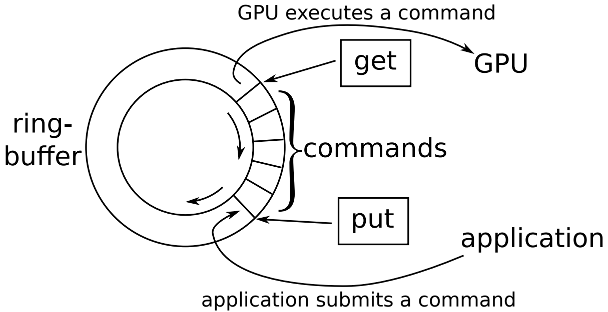


Device Control

- Devices are controlled through their device controller that accepts commands from the OS via it's device driver
- Devices are controlled through device registers and device memory
  - Control the device by writing device registers
  - Read status of device by reading device registers
  - Pass data to device by writing-/read results by reading device memory
- There are two ways to access device registers and device memory
  - Port-mapped I/O (PMIO)
    - Use special CPU instructions to access port-mapped registers and memory
    - e.g., x86 has different **`in`** and **`out`** commands that can transfer 1, 2 or 4 bytes between the CPU and a device
  - Memory-mapped I/O (MMIO)
    - Use the same *address space* for RAM and device memory
    - Some addresses map to RAM, others map to different devices
    - To access device registers/memory just access devices' memory region
  - Some devices use a hybrid approach and use both, PMIO and MMIO

Device Control: Nvidia General Purpose GPU

- Command submission channel: Ring-buffer and put/get device registers are memory-mapped
- Mapping can be exposed to application
  → Application can submit commands in user-mode



(Image by Mathias Gottschlag)

Summary

- The OS is an abstraction layer between applications and hardware
  - Multiplexes hardware and hides hardware details
  - Provides protection between processes/users
- The CPU provides a separation of User and Kernel Mode. These modes are **required** for an OS to provide protection between applications.
- CPU can execute commands faster than memory can deliver instructions and data
  - Memory hierarchy mitigates this memory wall
  - Memory needs to be carefully managed by the OS to minimize slowdowns
- Device drivers control hardware devices through PMIO or MMIO
- Devices can signal the CPU (and through the CPU notify the OS) through interrupts