

Betriebssysteme 1

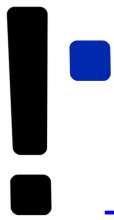
SS 2016

Prof. Dr.-Ing. Hans-Georg Eßer
Fachhochschule Südwestfalen

Foliensatz A:

- Einführung

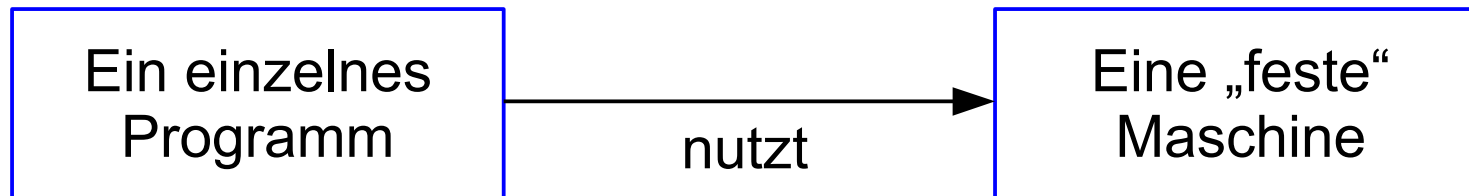
v1.0, 2016/03/22



Einführung und Motivation

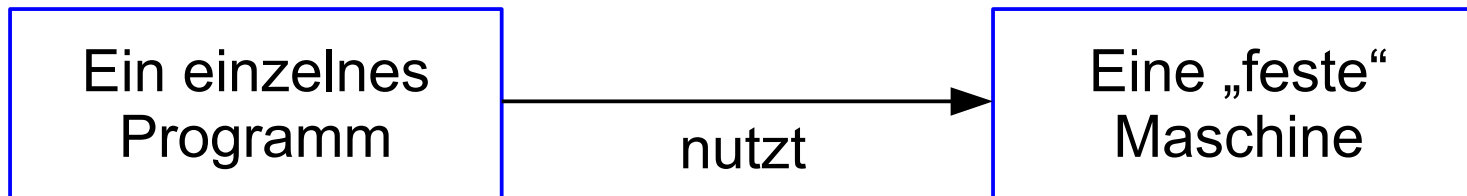
Betriebssysteme – in 2 Minuten (1)

- Beziehung Software $\leftrightarrow$ Hardware
- Einfache Variante:

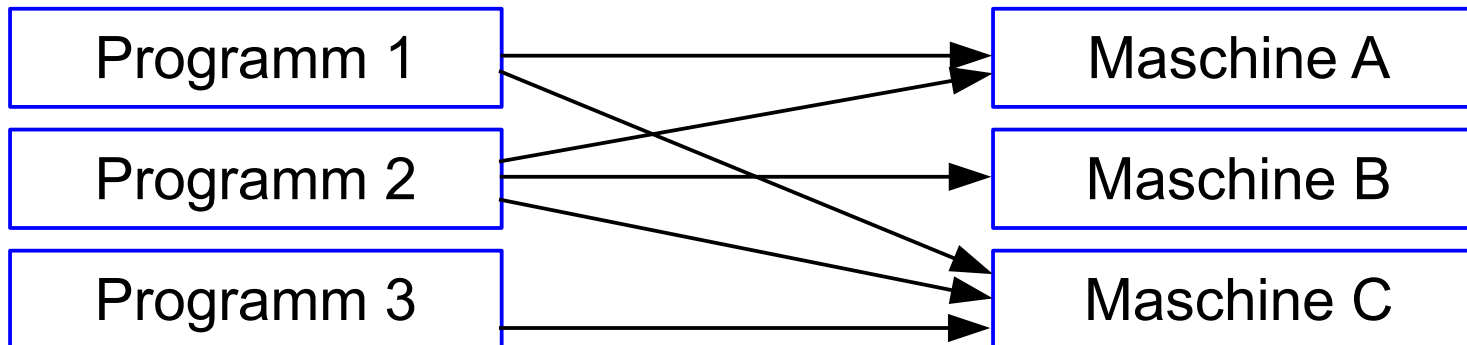


Betriebssysteme – in 2 Minuten (1)

- Beziehung Software $\leftrightarrow$ Hardware
- Einfache Variante:



- Probleme, wenn:



Betriebssysteme – in 2 Minuten (2)

Die Probleme im Detail

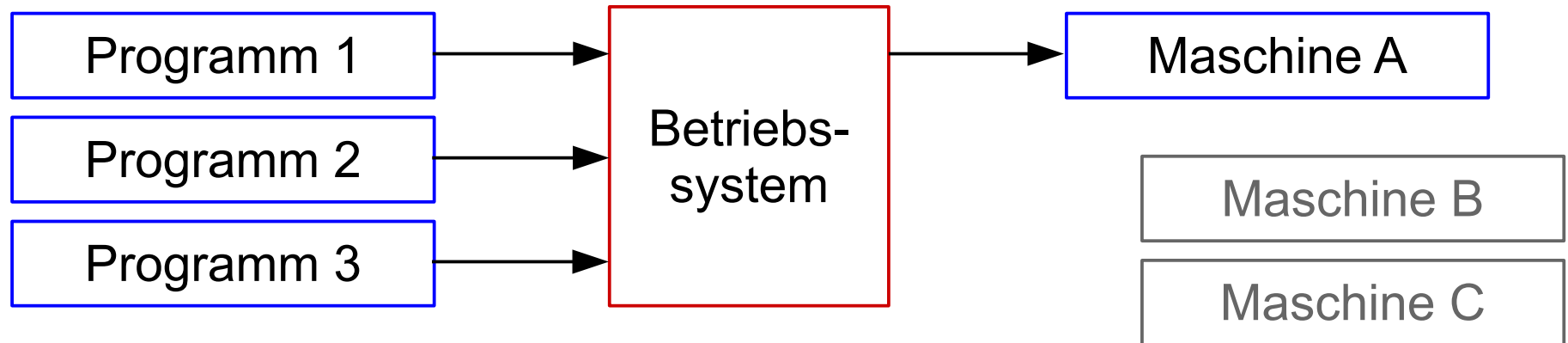
- Programm 1x entwickeln, soll aber auf verschiedenen Maschinen lauffähig sein
- Mehrere Programme sollen auf einer Maschine laufen → müssen sich die Ressourcen teilen

Programme so entwickeln, dass diese Probleme verschwinden? Schwierig...

Lösung: Betriebssystem

Betriebssysteme – in 2 Minuten (3)

- Betriebssystem zwischen Software und Hardware



Fragen zu Betriebssystemen

- Wie benutzen? → Endanwender-Perspektive
- Wie installieren, konfigurieren, absichern?
→ Administrator-Perspektive
- Wie das richtige BS zu gegebener Hardware /
bestimmten Anforderungen auswählen?
→ IT-Entscheider-Perspektive
- Wie Programme entwickeln, die (gut!) auf einem
bestimmten BS laufen? → Systemprogrammierung
- Wie ein BS entwickeln?

Aufgaben von Betriebssystemen (1)

- Abstraktionsschicht zwischen Hardware und Programmen (→ virtuelle Maschine)
- Verwaltung der vorhandenen Ressourcen
- Einheitlicher Zugriff auf Geräte einer groben Kategorie, z. B.:
 - ♦ *Datenträger* (Plattenpartition, CD, DVD, Diskette, USB-Stick, Netzwerk-Volume)
 - ♦ *Drucker* (PostScript-Laser, Etikettendrucker, Billig-Tintenstrahler, ...)

Aufgaben von Betriebssystemen (2)

- Schützt Hardware vor direkten Zugriffen
(→ defekte oder bösartige Software)
- Befreit Software vom Zwang, die Hardware im Detail zu kennen
- Zulassen mehrerer Benutzer und Abgrenzung
(Multi-user)
- Parallelbetrieb mehrerer Anwendungen
(Multi-tasking): faire Aufteilung der Ressourcen

Aufgaben von Betriebssystemen (3)

- Virtualisierung des Speichers
 - Anwendungen müssen nicht wissen, wo sie im Hauptspeicher liegen
 - Speicher über phys. RAM hinaus verfügbar (Swap etc.)

Beispiele (1)

Desktop-PC – die Standardaufgabe, Intel & Co.

- Anwendungsprogramme (Office, Grafik, kaufmännische Software etc.)
- Internet-Zugang und Web-basierte Anwendungen (WWW, E-Mail, File Sharing, ...)
- Datenbank-Client
- Software-Entwicklung
- Multimedia



Beispiele (2)

Server-PC

Häufig ähnliche Hardware wie Desktop-PC, aber ganz andere Einsatzgebiete:

- Web- / FTP- / Mail-Server
(Internet oder Intranet)
- Datenbank-Server
- „Number Crunching“ bzw.
High Performance Computing (oft: Cluster)

Beispiele (3)

Industrieanwendungen

- Robotersteuerung
- automatische Navigation
- Temperaturregelung
- Motorenkontrolle
- Herzschrittmacher

→ **Echtzeit-Betriebssysteme**
(real time operating systems)

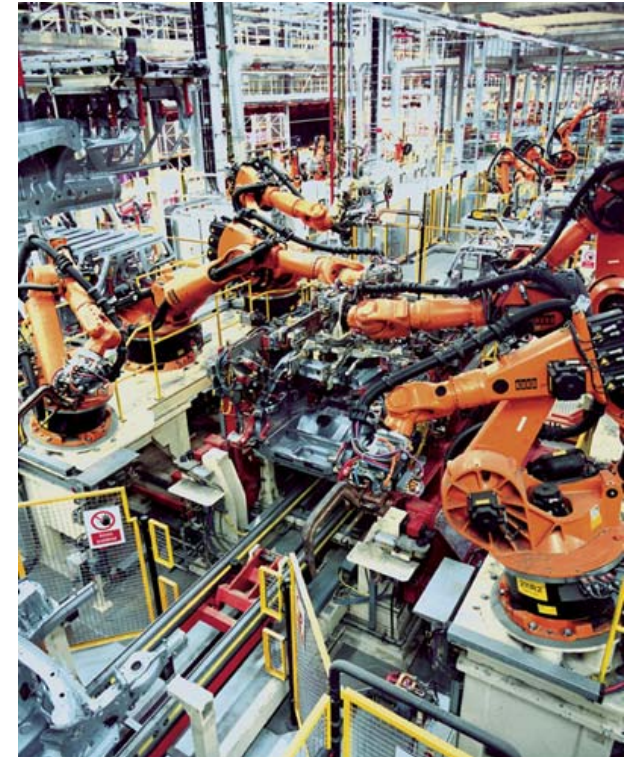


Bild: Wikipedia, KUKA Schweißanlagen

Beispiele (4)

Embedded systems (ohne Echtzeit-Ansprüche)

- Mobiltelefone, einfache mobile MP3/Video-Player
- Fernseher, DVD-Player
- Raspberry Pi & Co.
- DSL-WLAN-Router (mit Firewall etc.)
- Taschenrechner
- Videospiel-Konsolen
- Geldautomaten

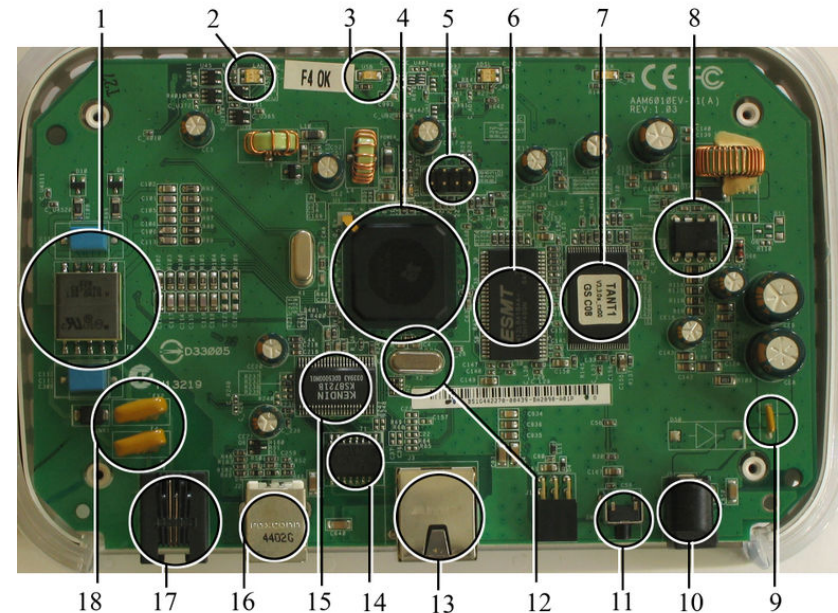


Foto: Wikipedia
(Mike1024)

keine Beispiele sind:
– Blu-ray-Player, Smart TV
– Smartphone, Tablet

Software-Entwicklung (1)

Beim Programmieren tauchen häufig Probleme in zwei Bereichen auf:

- **Zuverlässigkeit**

Software tut nicht das, was sie soll;
unerwartetes Verhalten;
mangelnde Fehlertoleranz

- **Sicherheit**

Software ist nicht geschützt vor Angriffen
durch Dritte

Software-Entwicklung (2)

Funktionsweise des Betriebssystems nicht klar
→ fehlerhaft programmierte Anwendungen, z. B.

- Race Conditions
- Buffer Overflows

Darum verstehen und lernen, wie
Betriebssysteme intern arbeiten



„Betriebssysteme“ an der FH SWF

Hans-Georg Eßer

- Dipl.-Math. (RWTH Aachen, 1997)
Dipl.-Inform. (RWTH Aachen, 2005)
Fachjournalist (FJS Berlin, 2006)
Dr.-Ing. (Univ. Erlangen-Nürnberg, 2015)
- Chefredakteur Linux-Zeitschrift (seit 2000)
und Autor diverser Computerbücher
- 2006-2016 Dozent an verschiedenen Hochschulen:
Betriebssysteme, Rechnerarchitektur, IT-Infrastruktur,
Informatik-Grundlagen, Systemprogrammierung,
Betriebssystem-Entwicklung, IT-Sicherheit
- seit 2016 Professor für Betriebssysteme an der FH Südwestfalen



Betriebssysteme 1 bis 3

- **Betriebssysteme 1:**
allgemeine Einführung, theoretische Grundlagen
- **Betriebssysteme 2:**
Fokus auf Linux-Administration
- **Betriebssysteme 3:**
Fokus auf Microsoft-Server-Administration

Zur Veranstaltung (1)

Veranstaltung kombiniert:
Theorie und Praxis der Betriebssysteme

Service / Web-Seite: <http://swf.hgesser.de>

- Vorlesungsfolien und ergänzende Literatur / Aufgabenblätter für das Selbststudium
- Vorlesungs-Videos
(*aber*: Besuch der Vorlesungen dringend empfohlen!)
- Probeklausur gegen Semesterende

Zur Veranstaltung (2)

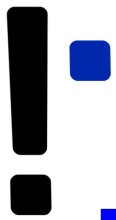
Hilfreiche Vorkenntnisse:

- **Linux-Shell** – Benutzung der Standard-Shell *bash* unter Linux
→ Bash-Crashkurs
- **C** – Grundlagen der Programmierung in C
(oder C++, C#, Java)
- **Rechnerarchitektur (1. Semester)**
→ grober Aufbau eines Computers
(Prozessor, Hauptspeicher, Peripherie etc.)

Zur Veranstaltung (3)

Betriebssysteme 1				
Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots
	180 h	6 CP	1. Sem.	Sommersemester
1	Lehrveranstaltungen Vorlesung: 4 SWS / 45 h	Kontaktzeit 4 SWS / 45 h	Selbststudium 135	

- **Selbststudium:** Linux-Praxis (Administration) und Programmierübungen zur Theorie-Vertiefung unter Linux; Literatur und Übungen folgen



Kurze Demo der Debian-VM

Zur Veranstaltung (4)

Fragen:

- direkt in der Vorlesung (Handzeichen)
- oder danach oder per E-Mail

Zur Veranstaltung (5)

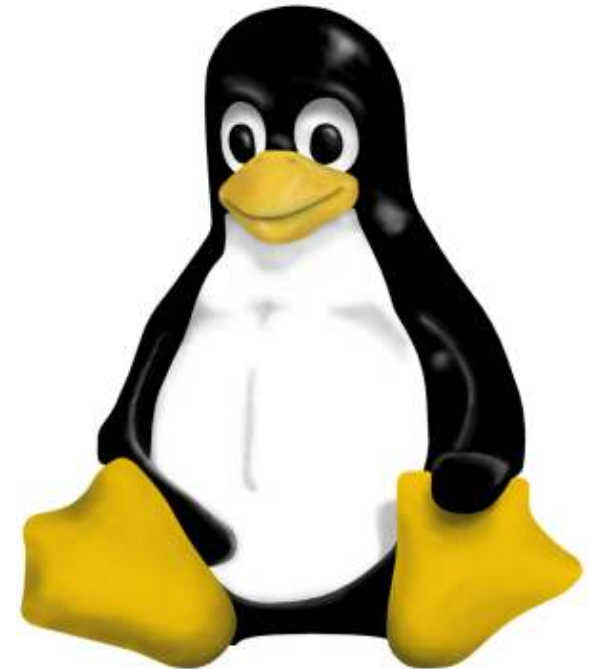
Linux-Administration

- Nutzen von Shell-Befehlen
- Standard-Datei- und -Verzeichnis-Operationen
- Editor vi
- Shell-Variablen, Unix-Filter-Programme
- Jobs und Prozesse
- Software-Verwaltung
- Einrichten von Partitionen
- Dateisuche, Auskunft
- Benutzer- und Gruppen-Rechte
- keine grafischen Tools – auch wenn es welche gibt
- stattdessen: Kommandozeilentools, Konfigurationsdateien, Shell-Skripte
- verstehen, was im Hintergrund abläuft

Theorie

- Im Theorieteil nicht: „Wie bediene ich ... ?“, sondern:
„Wie und warum funktioniert ... intern?“
- Konsequenzen für Anwendungsentwickler
- Sicherheitsprobleme
- Auswahl eines geeigneten Betriebssystems

- Etabliertes Standardsystem für sehr viele Plattformen (PC Desktop / Server, Embedded etc.)
- vor allem auf Servern weit verbreitet
- Offene Kernel-Quellen:
 - nachlesen, wie etwas geht
 - ändern, was nicht gefällt
- praktische Übungen: VirtualBox-VM mit Linux (oder Installation auf echtem Rechner)



Literatur: BS Praxis / Linux

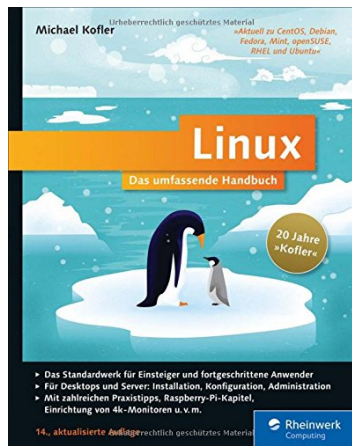


Grundlagenbuch Linux

Grundlagen, Techniken, Lösungen
(Eßer, Dölle)

Data Becker, 2007

→ als PDF-Dokument über Webseite

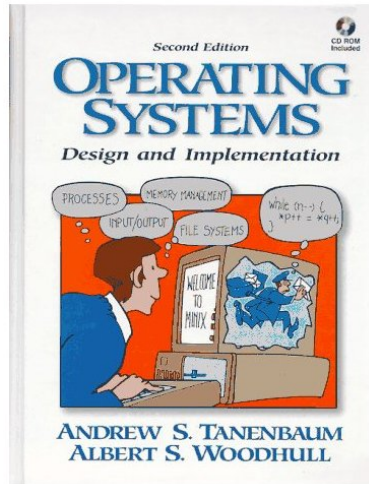


Linux: Das umfassende Handbuch (Kofler)

Rheinwerk Computing, 2015

49,90 €

Literatur: BS Theorie (1)

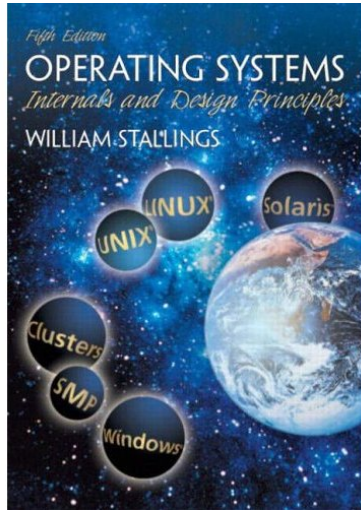


Operating Systems
Design and Implementation
(Tanenbaum, Woodhull)
Prentice Hall
(englisch)

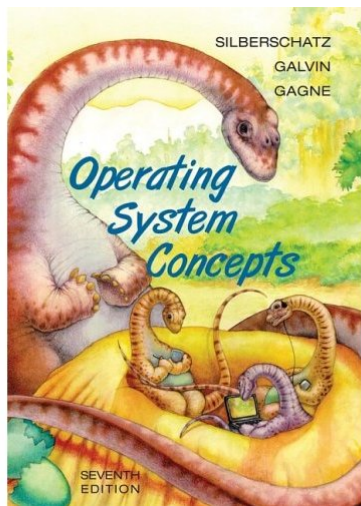


Betriebssysteme
Ein Lehrbuch mit Übungen zur System-
programmierung in Unix/Linux (Ehses et al.)
ISBN 3-8273-7156-2
Pearson Studium, 30 Euro

Literatur: BS Theorie (2)



Operating Systems
Internals and Design Principles
(Stallings)
Prentice Hall, ca. 80 Euro
(englisch)



Operating System Concepts
(Silberschatz, Galvin, Gagne)
Wiley, ca. 52 Euro
(englisch)



Gliederung

- *A: Einleitung*
- B: Prozesse und Threads
- C: Geräte und Interrupts
- D: Scheduler
- E: Synchronisation und Deadlocks
- F: Speicherverwaltung
- G: Dateisysteme
- H: Inter-Prozess-Kommunikation