Vorlesung Betriebssysteme I

Thema 1: Einführung

Robert Baumgartl

5. Oktober 2015

Organisatorisches

- 2/0/2, d. h., 90’ Vorlesung und 90’ Praktikum pro Woche
- Vorlesung dienstags 15:10 Uhr, Z 211
- Lehrender: Prof. Robert Baumgartl
- Kontakt:
  - robert.baumgartl@informatik.htw-dresden.de
  - dem Subject bitte „`[BS1]`“ voranstellen
  - Tel. 462 2510
  - Raum Z 357
- Praktikum in Linux-Laboren (Z 136c/Z 146a)
  - Betreuung durch mich und Laboringenieure (Herr Schubert, Herr Paul)
  - Start: **12. 10. 2015**
- Prüfung: Klausur, 90’, Voraussetzung: Beleg

Vorkenntnisse

Bitte um Handzeichen – Wer hat schon

- mit Windows gearbeitet?
- mit Linux (o. a. Unix) gearbeitet?
- einen der Editoren `vi` oder `emacs` genutzt?
- in C programmiert?
- `make` eingesetzt?
- `fork()` beim Programmieren benutzt?
- in der Open-Source-Community mitgewirkt?
- einen Treiber geschrieben?

Vorkenntnisse II

Wer weiß, was das macht:

```
char *foo(char *dest, const char *src)
{
    while(*dest++ = *src++);
}
```

. . . und das?

```
bash$ :(){ :|:&};:
```

(Vorsicht! Nicht ausprobieren!)

Wozu befassen wir uns mit Betriebssystemen?

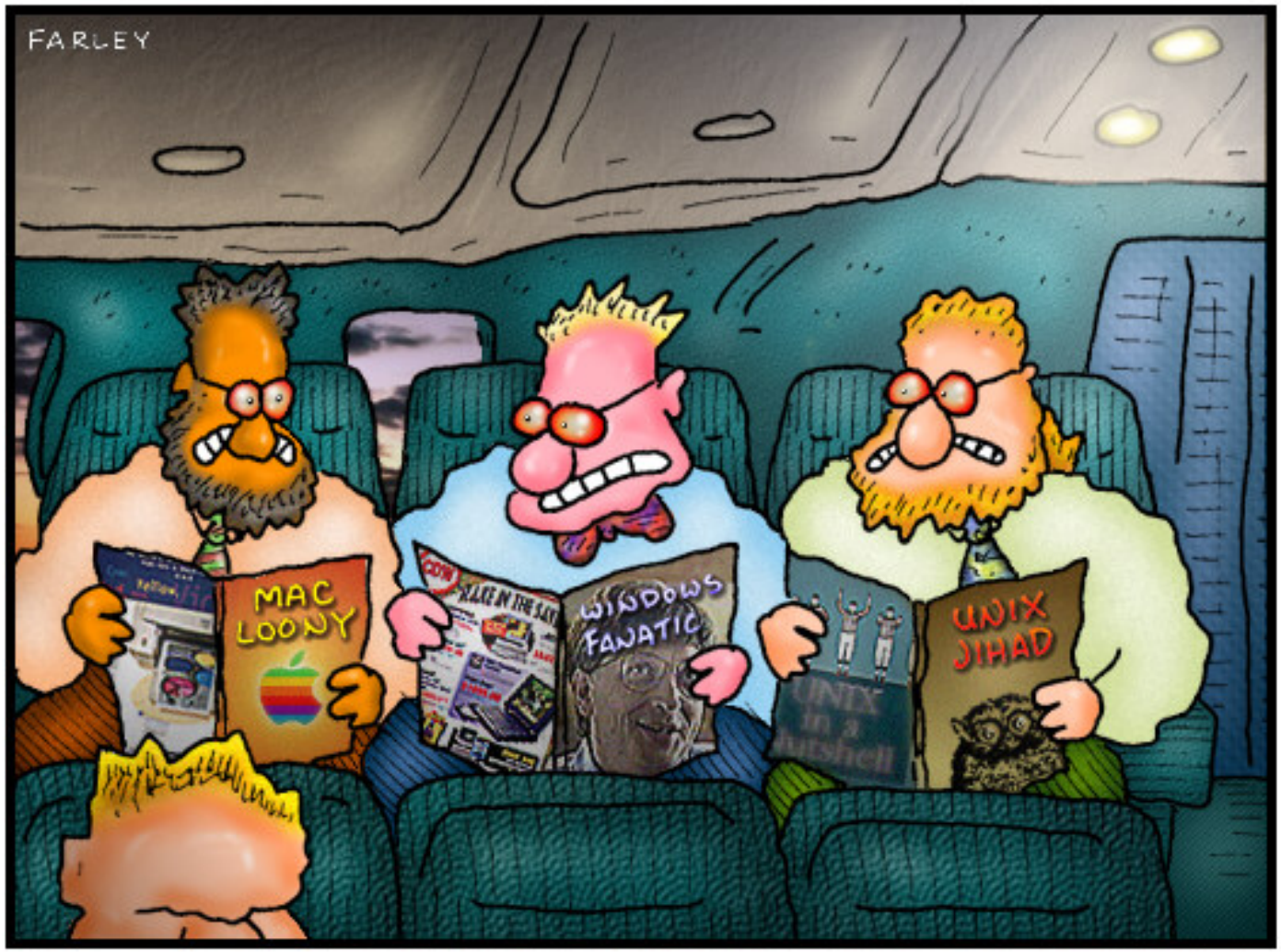
... es gibt doch Windows!

Einige Gedanken:

- Grundlagenfach der Informatik
- BS gehören zu den komplexesten Softwaresystemen überhaupt!
- aktiver Forschungsgegenstand
  - Betriebssysteme-Sicherheit
  - Skalierbarkeit
  - Sensornetze
  - Echtzeitbetriebssysteme
- Linux!
- und zuguterletzt: *Wir wollen doch kompetent die Frage beantworten, ob Linux oder Windows besser ist?!*

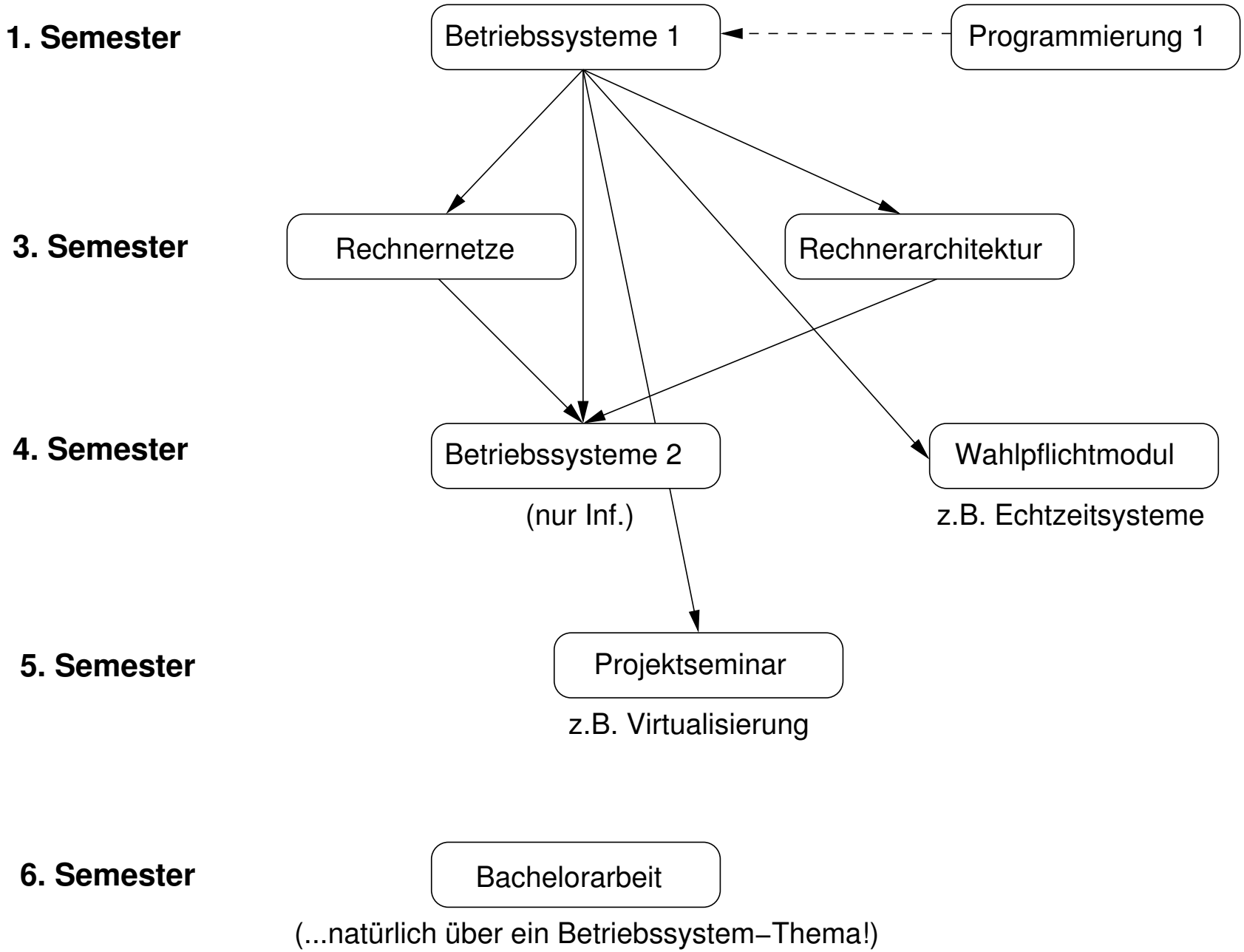
DOCTOR FUN

20 Dec 96



The beginning of a very long flight

Einordnung der Lehrveranstaltung



Vorläufige Themenübersicht

1. Einführung
2. Linux in a Nutshell
3. Dateisystem
4. Grundlegende Begriffe, Teil 2
5. Aktivitäten
6. Kommunikation
7. Scheduling
8. Threads (Aktivitäten, die zweite)
9. Synchronisation

Einige Aspekte von Betriebssystemen

- Bedienung
- Administration
- Programmierung für Betriebssysteme
- Programmierung von Betriebssystemen
- Abstraktionen für Aktivitäten (Prozesse, Threads, Coroutinen, Fibers, Tasks)
- Fehlertoleranz
- Security & Safety
- Forensik

Ziel des Kurses

Vermittlung von vorwiegend praktischen Kenntnissen zu

- Nutzung elementarer Werkzeuge
- Automatisierung von Bedienhandlungen
- Interaktion zwischen Applikationen und Betriebssystem(en)
- Struktur und Vorgängen innerhalb von Betriebssystemen
- Unix-artigen und Windows-Betriebssystemen

Eine kurze Geschichte der . . . Betriebssysteme

Andrew Tanenbaum unterscheidet 4 Epochen

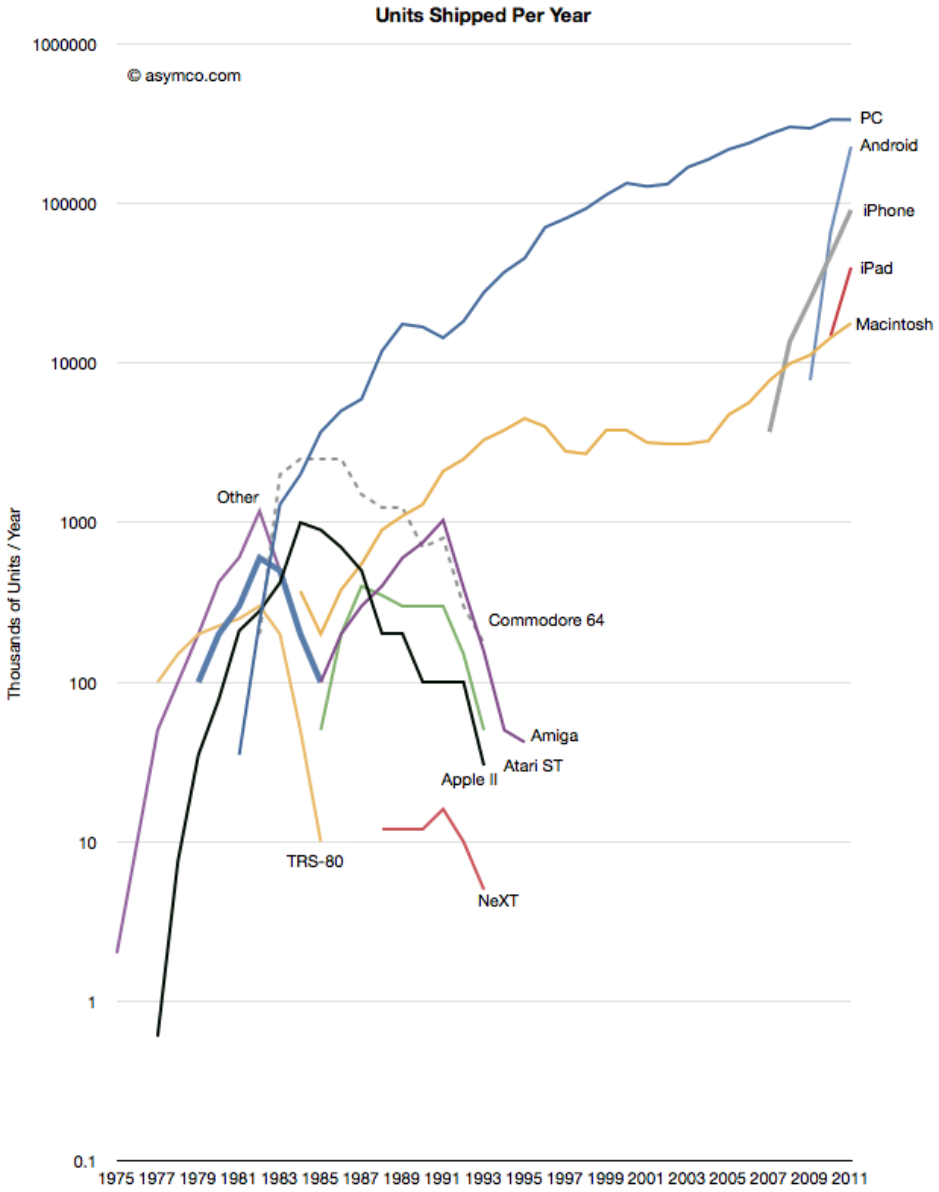
1. 1945-55 – Elektronenröhren
   - keine Betriebssysteme
2. 1955-65 – Transistoren
   - Mainframes – kommerzielle Computer
   - Batchsysteme (Ziel: maximale Rechnerauslastung)
3. 1965-75 – niedrig integrierte Schaltkreise
   - IBM System/360 $\rightarrow$ **OS/360** (Ziel: Kompatibilität)
   - Multiprogramming (mehrere Ausführungseinheiten gleichzeitig)
   - Spooling
   - Timesharing (Ziel: Reduktion der Systemantwortzeit)
   - **MULTICS** (ambitioniert, aber erfolglos)
   - Minicomputer (kleiner als Mainframe; DEC PDP-1. . . -11)
   - **UNIX** (portabel, offen, kollaborativ entwickelt)

Eine kurze Geschichte der . . . Betriebssysteme

Andrew Tanenbaum unterscheidet 4 Epochen

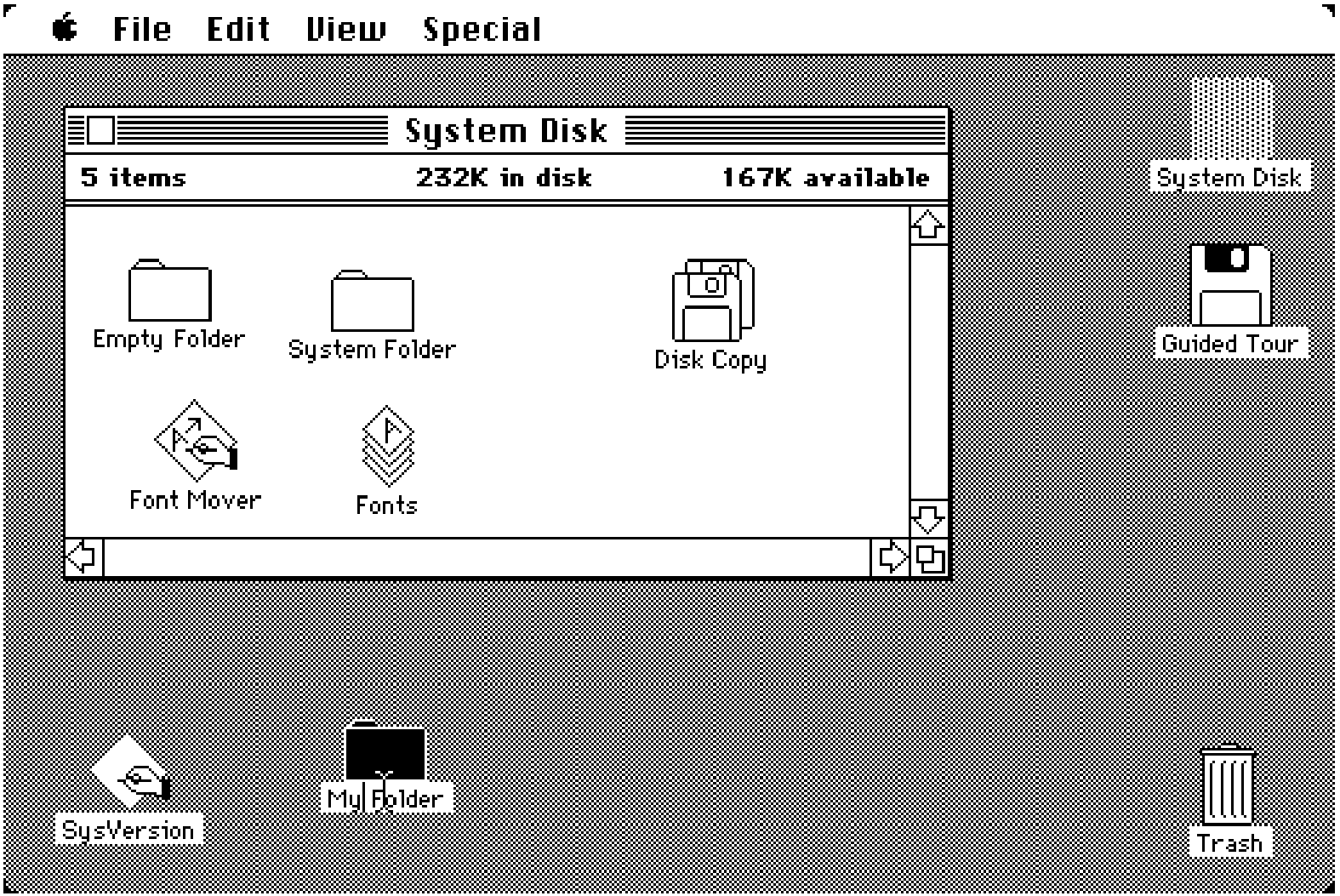
4. 1975-heute — hoch integrierte Schaltkreise
   - 1976: Prozessor Zilog Z80 $\rightarrow$ **CP/M**
   - etwa ab 1977: Homecomputer (Apple II, C64, . . .)
   - 1979: Prozessor i8088; PC
   - 1980: **QDOS** $\rightarrow$ **MS-DOS**
   - 1984: Apple Macintosh $\rightarrow$ **MacOS** (GUI)
   - 1985: **Microsoft Windows 1.0**
   - 1992: **Linux**

Plattformen fürs „Personal Computing“



- verkaufte „Personal Computing Units“ pro Jahr
- Quelle: *http://twitpic.com/87nbjj*

GUI von MacOS (1984)



(http://upload.wikimedia.org/wikipedia/en/5/50/Apple_Macintosh_Desktop.png)

Betriebssysteme

Es gibt:

- Windows-Familie (2.0 $\rightarrow$ 10)
- Linux
- MacOS X

... das wars, oder?

- MS-DOS, RTEMS, QNX, FreeBSD, SymbianOS, PalmOS, RTAI, HP-UX, BeOS, Minix, Chorus, L4, Mach, Amoeba, OS/390, DCP, TOS, CP/M, VMS, eCos, Contiki, OS/2 ...
- vgl. *http://de.wikipedia.org/wiki/Liste_der_Betriebssysteme*

Betriebssysteme

Es gibt:

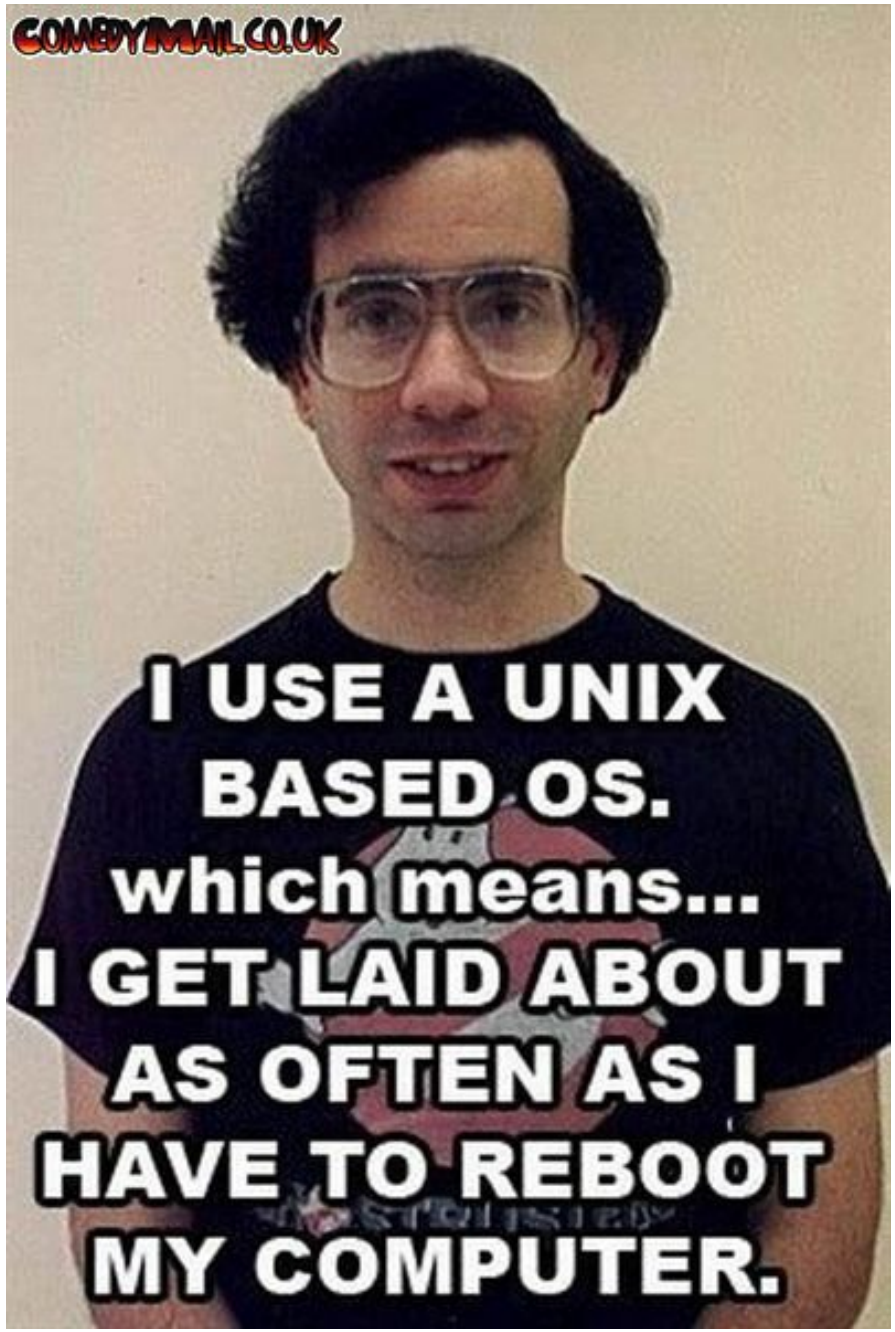
- Windows-Familie (2.0 $\rightarrow$ 10)
- Linux
- MacOS X

... das wars, oder?

- MS-DOS, RTEMS, QNX, FreeBSD, SymbianOS, PalmOS, RTAI, HP-UX, BeOS, Minix, Chorus, L4, Mach, Amoeba, OS/390, DCP, TOS, CP/M, VMS, eCos, Contiki, OS/2 ...
- vgl. *http://de.wikipedia.org/wiki/Liste_der_Betriebssysteme*

UNIX

- . . . ist kein Betriebssystem, sondern eine ganze Familie
- Name ist ein Wortspiel aus dem Vorgänger „Multics“ und „unique“
- zusammen mit der Programmiersprache C in den 70er Jahren entwickelt
- einige Vertreter: *BSD, System V, Linux, HP-UX, AIX, Solaris, Minix
- sogar Microsoft hat ein UNIX entwickelt: XENIX (es ist aber schon lange tot)
- der zugehörige Standard heißt POSIX
- beliebt vor allem im Serverbereich (aber nicht nur!)
- Nutzer haben mit Vorurteilen zu kämpfen . . .


COMEDYMAIL.CO.UK
I USE A UNIX BASED OS.
which means...
I GET LAID ABOUT AS OFTEN AS I HAVE TO REBOOT MY COMPUTER.

Philosophie von UNIX

9 Grundsätze von Gancarz:

1. Small is beautiful.
2. Make each program do one thing well.
3. Build a prototype as soon as possible.
4. Choose portability over efficiency.
5. Store data in flat text files.
6. Use software leverage to your advantage.
7. Use shell scripts to increase leverage and portability.
8. Avoid captive user interfaces.
9. Make every program a filter.

(Mike Gancarz: The UNIX Philosophy, Digital Press, 1995)

Lizensierungsaspekte

2 grundlegende Ideen:

- *Closed Source*: Quellcode ist geheim, „Betriebsgeheimnis“, steht i. a. nur dem Hersteller zur Verfügung
- *Open Source*: Quellcode steht prinzipiell jedem zur Verfügung $\rightarrow$ kann modifiziert und weiterverteilt werden (und soll dazu ermuntern)

Kommerzielle Lizenzen

- kann (muss aber nicht) Einblick in Quellcode umfassen (z. B. für nichtkommerzielle Zwecke)
- erfordert meist Vertrag („End User License Agreement“ (EULA))
- typische EULAs sind im EU-Raum jedoch unwirksam (zum Glück)

Kosten für:

- Entwicklungswerkzeuge
- Bibliotheken (z. B. für Protokollstacks oder zum Debugging)
- Royalties: pro Installation auf Zielgerät
- (Schulung der Entwickler)

GNU General Public License (GPL)

- Richard Stallman, 1989
- Kurzform:
  1. Das Werk darf für beliebige Zwecke verwendet werden (auch kommerziell).
  2. Das Werk darf beliebig weitergegeben werden, kostenlos oder kostenpflichtig. Der Quelltext (auch eigener Modifikationen) ist mitzuliefern.
  3. Das Werk darf beliebig modifiziert werden.
  4. Es dürfen keine Einschränkungen an diesen Regeln erfolgen.
- enthält sog. starkes „Copyleft“: erzwingt die Weiterverbreitung von aus freien Werken *abgeleiteten* Werken $\rightarrow$ niemand kann die Verbreitung eines ursprünglich freies Werk verhindern („Virulenz“)
- wichtigste Open-Source-Lizenz
- Beispiele: Linux, eCos, GCC, emacs, vi

Nachteile der GPL

- untersagt das Vermischen von GPL-Code mit Code, der unter inkompatibler Lizenz steht (also alle closed source, aber auch freie Software)
- $\rightarrow$ Binärtreiber bestimmter Grafikkarten sind eigentlich illegal im Linux-Kern (geduldet; „tainted kernel“)
- erschwert die Migration zu freier Software, da in Unternehmen existierende kommerzielle Software nicht ohne weiteres in diese integriert werden kann
- Verletzungen werden verfolgt! (*gpl-violations.org*)

Wozu benötigen wir nun ein Betriebssystem?

1. Bereitstellen von Diensten und dafür notwendigen Abstraktionen (z. B. „Prozess“, „Datei“, „Gerätetreiber“ u. v. a. m.)
2. Ressourcenverwaltung inklusive Protokollierung
3. Koordinierung paralleler Abläufe
4. Basis für Schutz und Sicherheit

Klassifizierung von Betriebssystemen

Kriterium: Nutzeranzahl

- Single-User-BS
- Multi-User-BS

Kriterium: Anzahl unabhängiger Aktivitäten

- Single-Tasking-BS
- Multi-Tasking-BS

Kriterium: Kommunikation mit der Umwelt

- BS zur Stapelverarbeitung (Batchbetrieb)
- interaktives BS
- BS für autonome Systeme

Klassifizierung von Betriebssystemen II

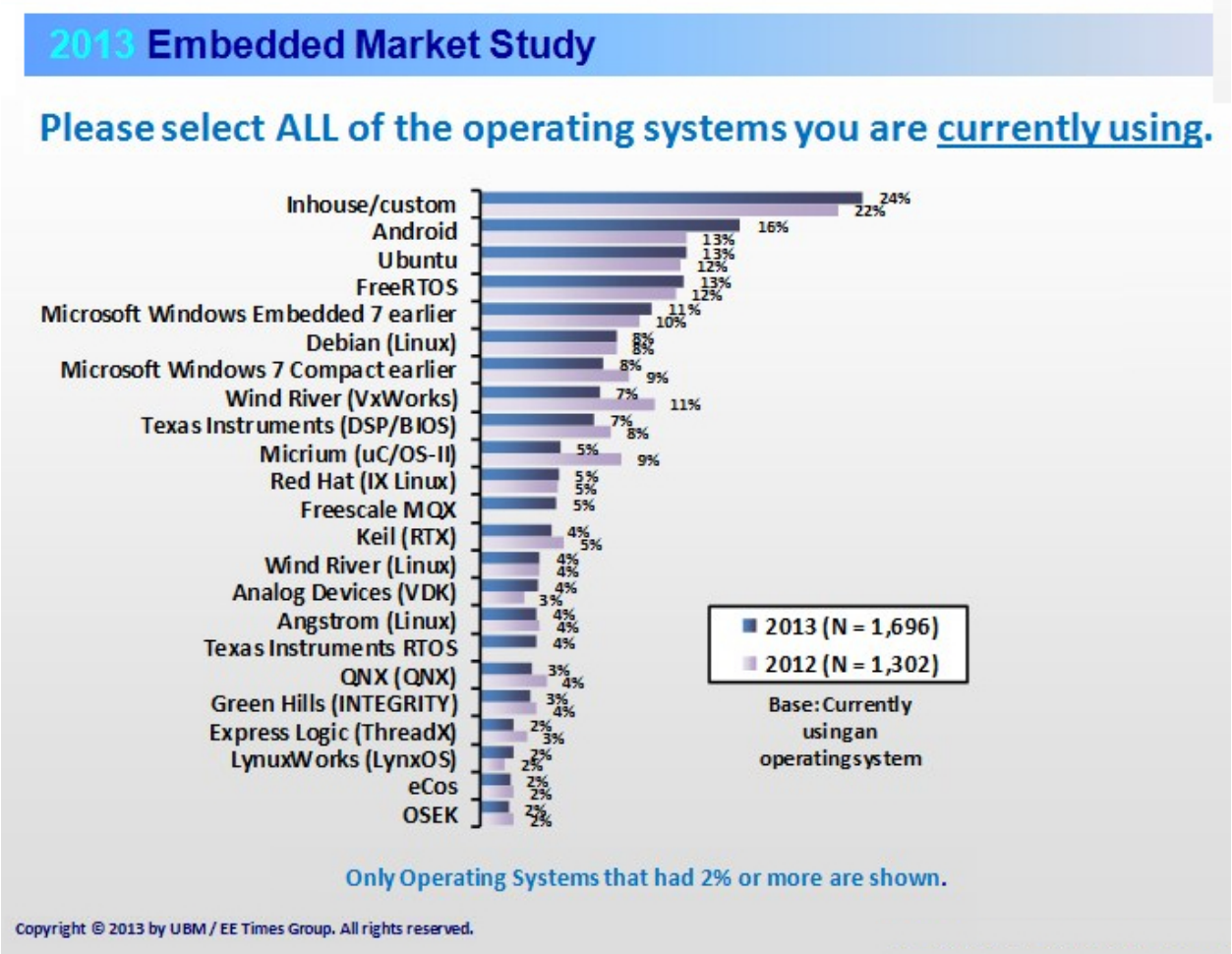
Kriterium: Verteilung

- lokales BS
- verteiltes BS

Kriterium: Zielarchitektur/Einsatzzweck

- Serverbetriebssystem
- eingebettetes Betriebssystem
- Echtzeitsystem
- Mainframe-BS
- BS für Personal Computer
- BS für Smart Card
- BS zur Ausbildung/Lehre

Apropos: welches Betriebssystem wird eingebettet eingesetzt?



Quelle: `http://www.eetimes.com/document.asp?doc_id=1263083`

Interaktion mit dem Betriebssystem

Paradigmen:

- vorwiegend textorientiert (Konsole, Shell, Eingabeaufforderung)
- grafische Oberfläche (Windows, KDE, Windowmaker)

Die Frage *Was ist besser?* führt unausweichlich zu Ärger

- keine Frage des Betriebssystems sondern der persönlichen Vorliebe

Modellierung und Strukturierung von BS

Problem: BS gehören zu den komplexesten Softwaresystemen überhaupt! $\rightarrow$ durch Lesen des Programmcodes kaum zu verstehen
Technik: Durch Reduktion der möglichen Kommunikationsbeziehungen zwischen Komponenten Übersicht schaffen.

Modell 1: Monolithisches System

- Andrew Tanenbaum: “The Big Mess”
- jede Routine, Funktion, . . . darf *jede* andere im System rufen $\rightsquigarrow$ unübersehbare Vielfalt potentieller Kommunikationsbeziehungen
- kein *Information Hiding*
- BS = Sammlung von Funktionen
- typisch für „historisch gewachsene“ Systeme
- effizient!

Modell 2: Geschichtetes System

- Kommunikation nur zwischen Instanzen benachbarter Schichten
- Beispiel: OSI-Schichtenmodell der ISO für Kommunikationsprotokolle (7 Schichten)
- leider kein vergleichbarer Standard in der BS-Technologie etabliert

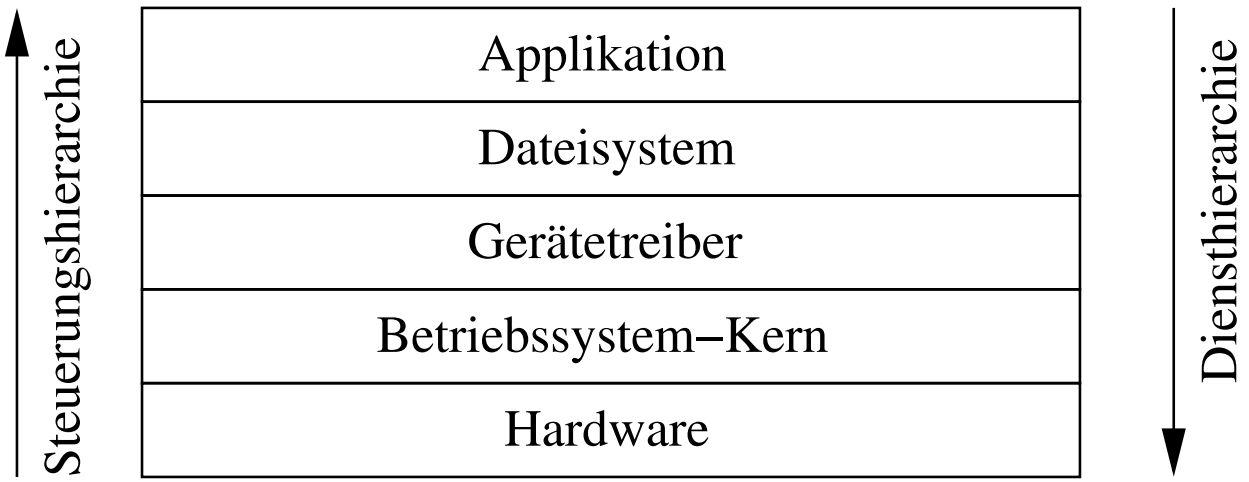


Abbildung: Beispiel für ein geschichtetes System

- Gefahr der Ineffizienz

Variante: quasikonsistente Schichtung

- Schichtung nicht zwingend:

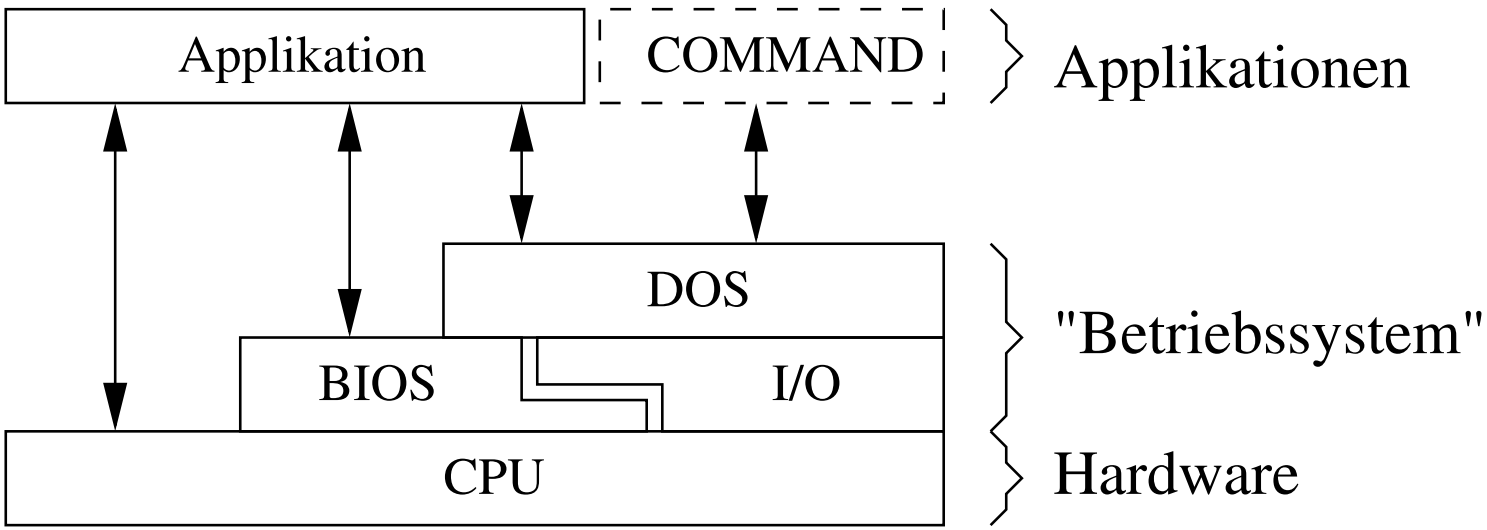
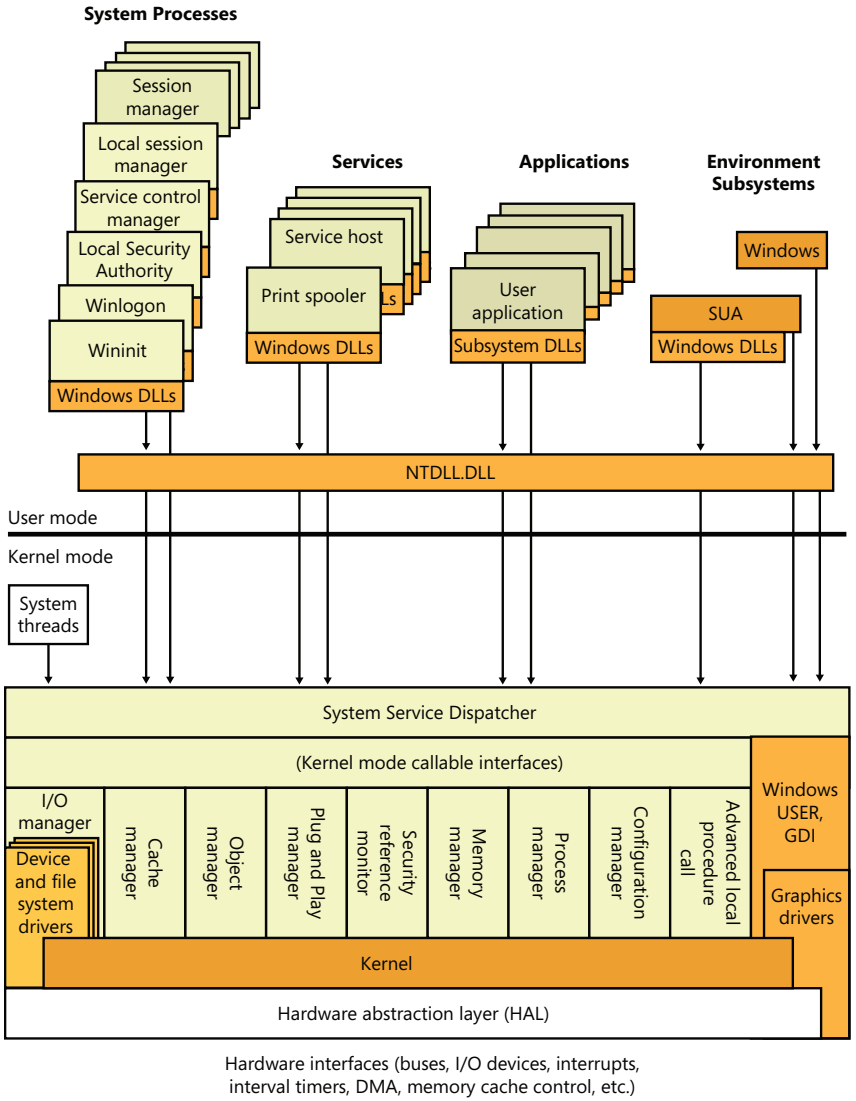


Abbildung: Quasikonsistente Schichtung im MS-DOS

Beispiel für ein komplex geschichtetes Modell

Mark Russinovitch et al: *Windows Internals.* 6th ed., Microsoft Press, 2012, S. 47



Modell 3: Client-Server-Modell

- Diensterbringung durch eine zentrale Instanz
- Client wendet sich mit Dienst-Wunsch an Server
- Server erbringt gewünschten Dienst, wenn möglich
- Beispiele: Speicherverwaltung im BS, NTP-Server, Drucker-Server, ...
- sog. Mikrokern-Architekturen wenden das Prinzip konsequent auf BS-Komponenten an

Zusammenfassung: Was haben wir gelernt?

1. Es gibt *viele* BS
2. Was ist UNIX?
3. Lizensierung: *Closed Source* vs. *Open Source*
4. Klassifikationskriterien von BS
5. Modellierung/Strukturierung von BS:
   - monolithisch
   - geschichtet
   - Client-Server-Beziehungen

Literaturvorschläge

- Andrew S. Tanenbaum: *Modern Operating Systems.* Pearson Education
- Richard Stallings: *Operating Systems.* Fifth Edition, Prentice-Hall
- Dokumentation der Geschichte des Windows-Betriebssystems: *http://www.winhistory.de/*
- Ellen Siever, Stephen Figgins, Robert Love, Arnold Robbins: *Linux in a Nutshell.* Sixth Edition, O'Reilly, 2009
- Cameron Newham: *Learning the Bash Shell.* Third Edition, O'Reilly, 2005