Kapitel V

Betriebssystem-Architekturen

Motivation (I)

Motivation (II)

- Betriebssystem-Entwicklung stellt sich aufgrund des großen Anforderungs-Umfangs und der hohen Anzahl an LOC als komplexe Aufgabe dar
- Zudem:

  Betriebssysteme weisen im Allgemeinen hohe Lebensdauer auf, sodass sie oftmals mehrere Jahrzehnte am Markt existieren
- Vor diesem Hintergrund:

  Beim Entwurf von Betriebssystem spielen Lesbarkeit (Verständlichkeit) des Quellcodes, Wartbarkeit und Erweiterbarkeit eine zentrale Rolle. Nur unter der Kenntnis verschiedener Architekturansätze und Einhaltung allgemeiner Architekturprinzipien können deshalb moderne, zukunftsfähige Betriebssysteme entwickelt werden.

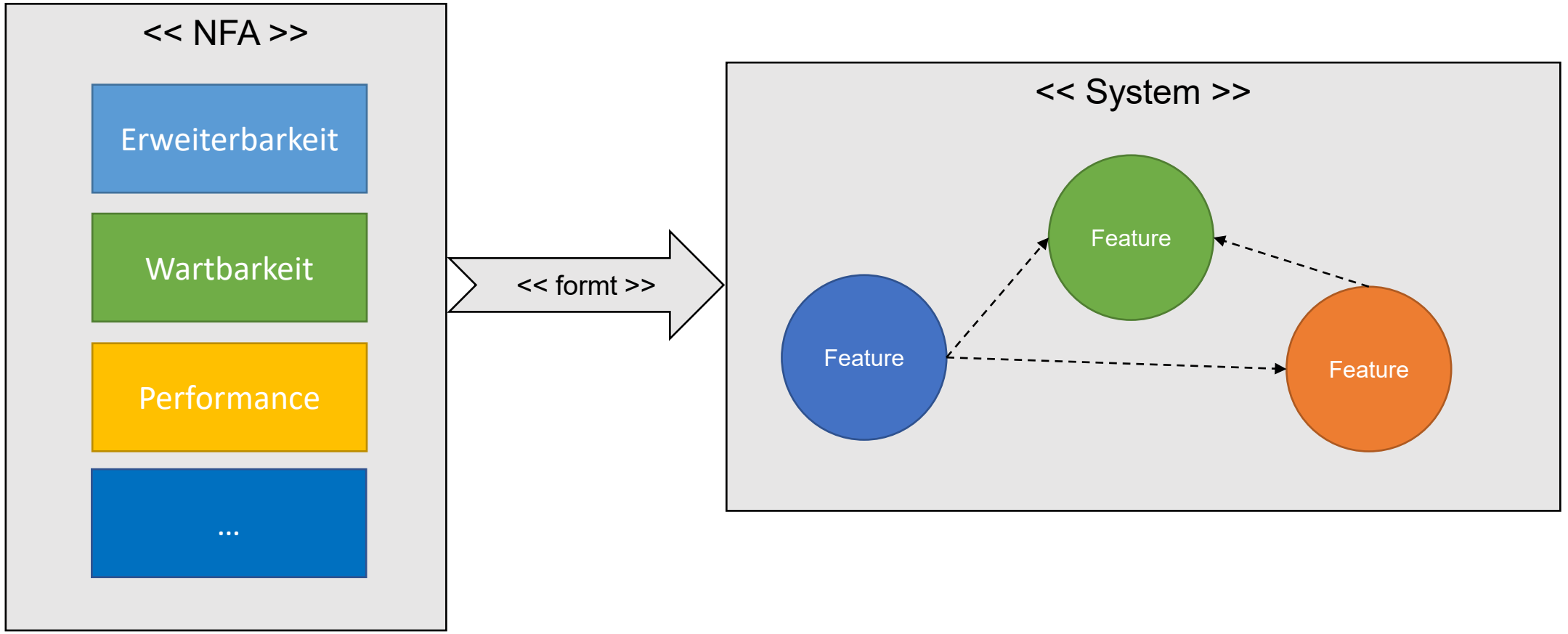
Merkmale guter Software-Architektur (I)

Was sind Merkmale guter Software-Architektur?

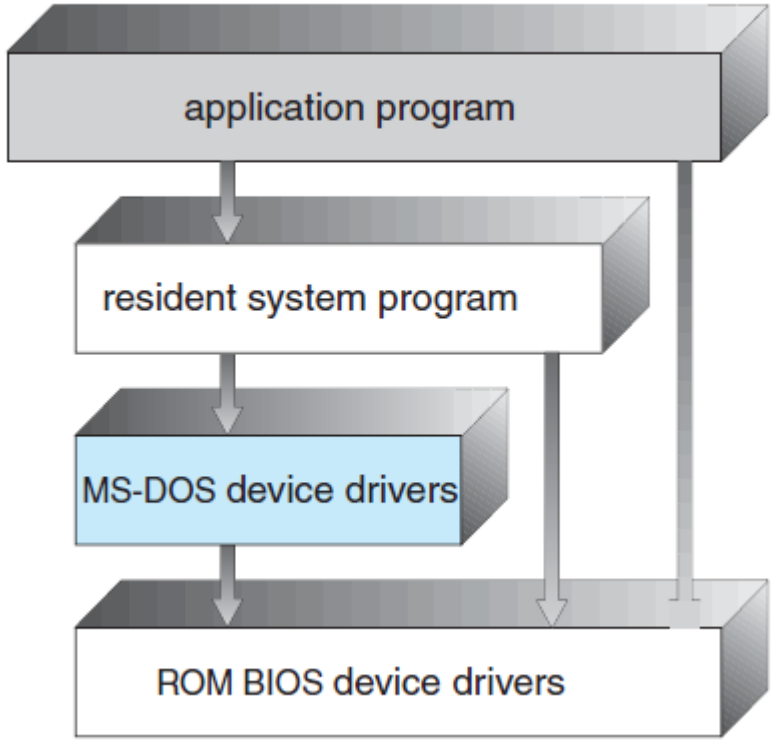
Merkmale guter Software-Architektur (II)

- Design for Change
- Design-Prinzipien:
  - SOLID
  - DRY
  - YAGNI
  - …
- Design Patterns
- Kohäsion vs. Kopplung (logischer, innerer Zusammenhang von Komponenten vs. äußere Abhängigkeiten)
- Getrieben durch Nicht-Funktionale Anforderungen bzw. Qualitätsanforderungen:
  Performanz, Wartbarkeit, Zuverlässigkeit, Sicherheit
- …

Merkmale guter Software-Architektur (III)



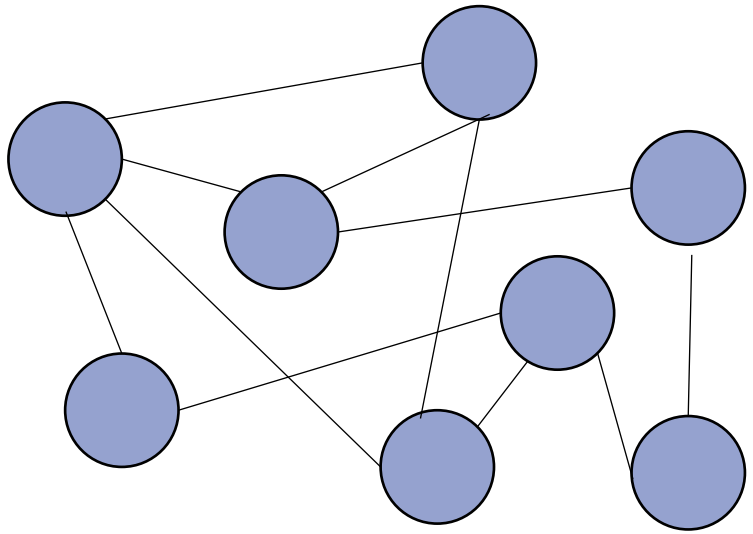
Einfache Architektur



MS-DOS Schichtenarchitektur:

- Schnittstellen und Schichten können übersprungen werden: Benutzerprogramme sind
  z. B. in der Lage, I/O direkt anzusprechen, Inhalte direkt auf dem Monitor anzuzeigen und Festplatten anzusteuern
- Dadurch: Abstraktion und Zugriffssicherheit nur geringfügig vorhanden
- Fehleranfällige Architektur, da ein nicht-funktionierendes Benutzerprogramm das gesamte System zum Absturz bringen kann

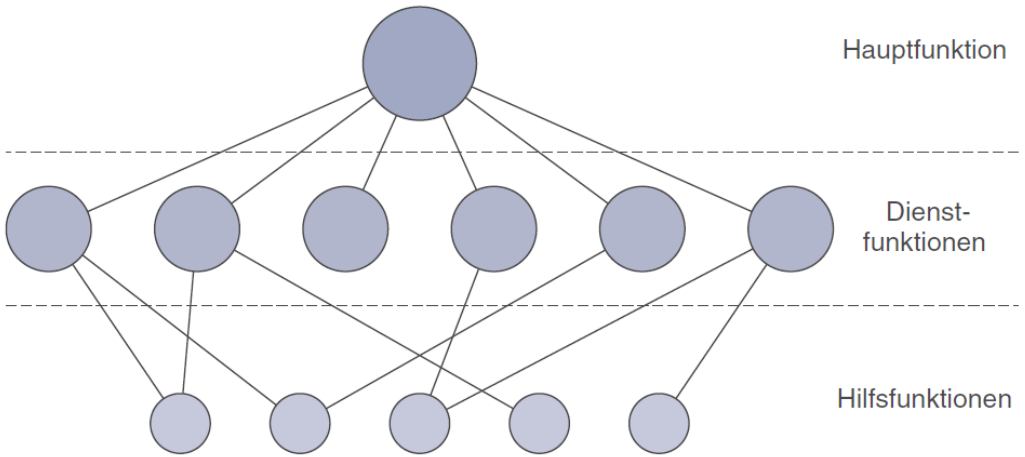
Monolithische Architektur (I)



Monolithische Architektur:

- Gesamtes System läuft als ein einziges Programm im Kernmodus
- Betriebssystem ist als eine Menge von Prozeduren realisiert
- Jede Prozedur darf mit allen anderen Prozeduren des Systems kommunizieren
- Dadurch:
  - hohe Komplexität durch großen Abhängigkeitsgraphen
  - Geringe Wartbarkeit und Verständlichkeit

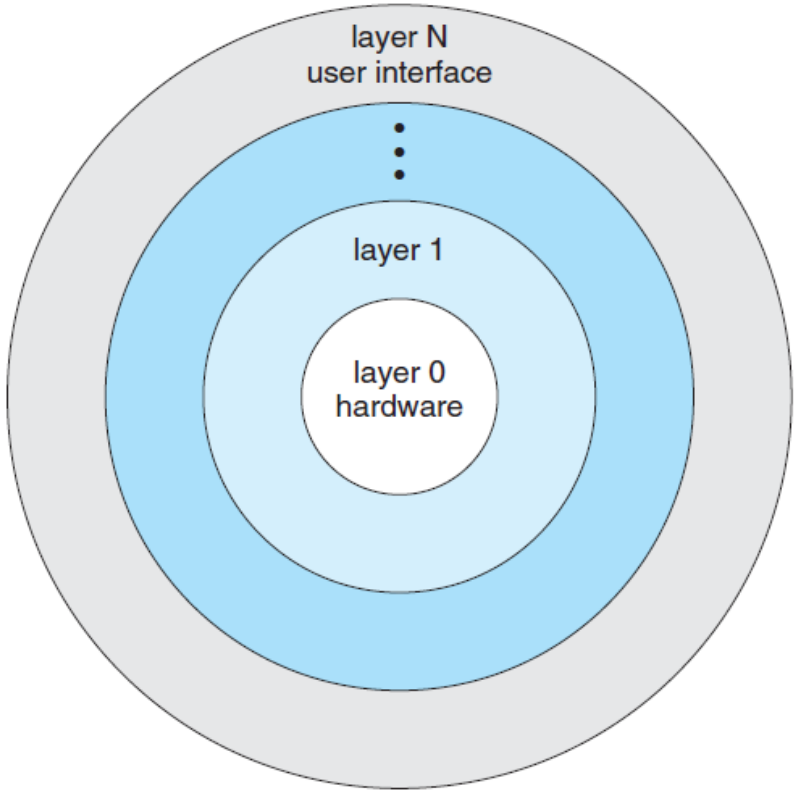
Monolithische Architektur (II)



Monolithische Architektur:

- Besser: Logische Unterteilung / Anordnung der Prozeduren in Schichten
- Hauptfunktionen rufen benötigte Dienstfunktionen auf
- Dienstfunktionen führen Systemaufrufe aus
- Hilfsfunktionen unterstützen die Dienstfunktionen

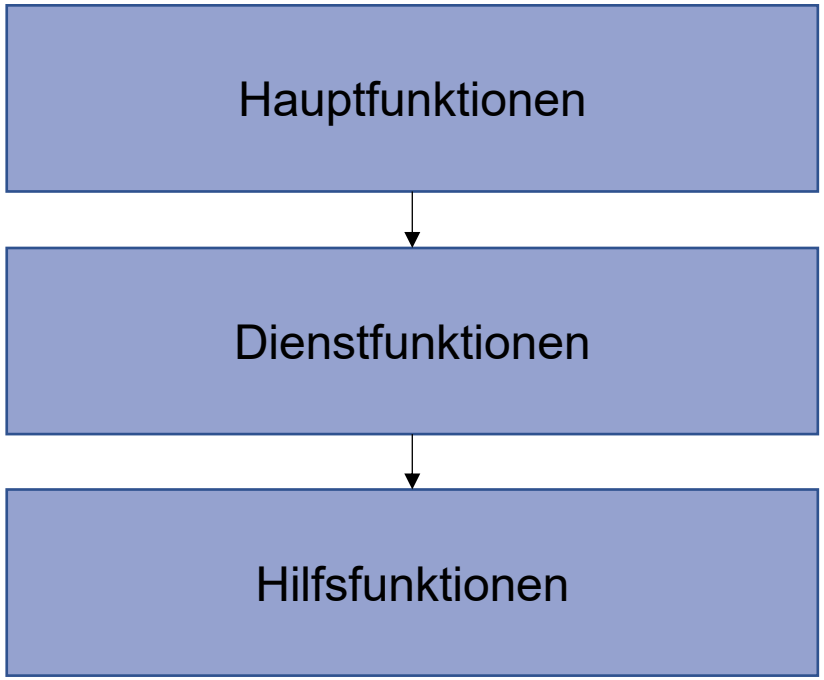
Schichten-Architektur (I)



Schichten-Architektur nach [AS12]:

- Pro Layer:
  - Benötigte Datenstrukturen
  - Menge von Funktionen
- Schicht *n+1* ruft Funktionalitäten der Schicht *n* auf
- Schicht *n* stellt Abstraktionen im Sinne einfacher Schnittstellen für Schicht *n+1* bereit
- Damit: Transparenz in den Aufrufen, Trennung der Zuständigkeiten, Komplexitätsreduktion
- Aber auch: Performance-Loss durch Indirektion
- Eigentlich bekannt als: Onion-Architektur / Schalenmodell / Ring-Architektur

Schichten-Architektur (II)

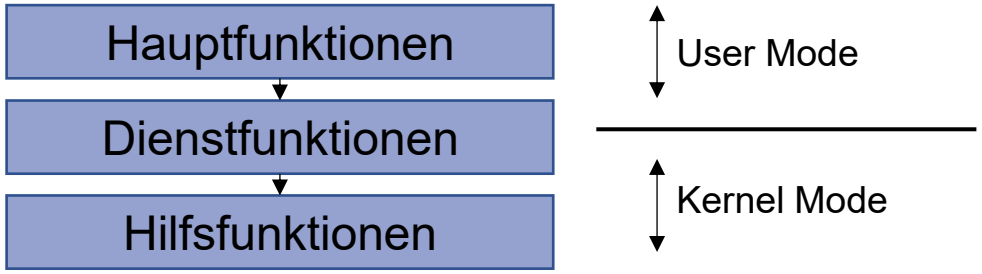


Schichten-Architektur nach [MB17]:

- Im Gegensatz zu der Interpretation von [AS12] tatsächliches Schichten- und nicht Schalen-Modell
- Orientiert sich zunächst an monolithischer Architektur, führt aber Schichten zur logischen Unterteilung der Funktionen ein
- Je näher eine Schicht an der Hardware, desto privilegierter die darin enthaltenen Berechtigungen

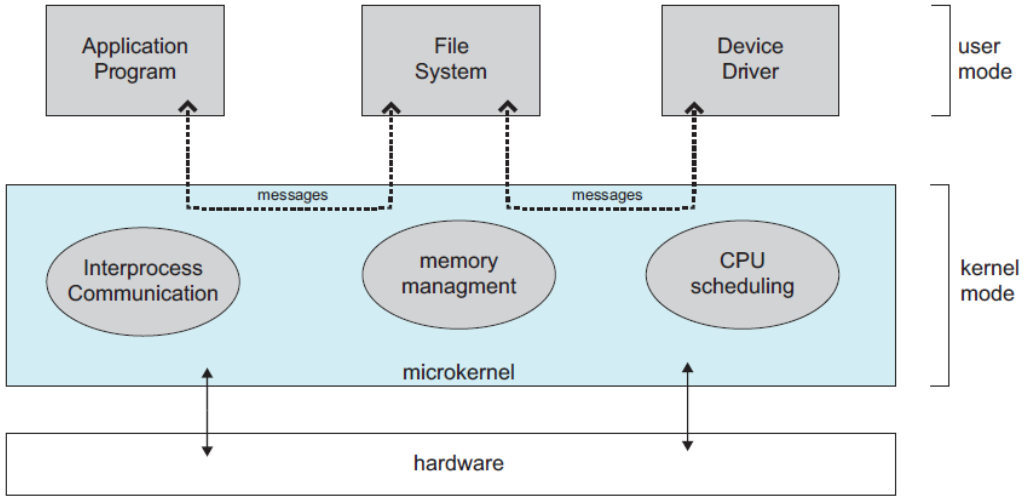
Mikrokern-Architektur (I)

- Bei bisher vorgestellten Architekturen:
  Entwickler wählen „frei“, wo Grenze zwischen Kern- und Benutzer-Modus liegt:



- Dabei:
  Traditionell Ausführung aller Schichten im Kern-Modus
- Dadurch:
  Fehler, die im Kern-Modus auftreten, können das gesamte System zum Absturz bringen
- Mikrokern-Architektur adressiert dieses Problem, indem so wenig wie möglich Funktionen im Kernmodus ausgeführt werden

Mikrokern-Architektur (II)



Mikrokern-Architektur:

- Kernel-Funktionsumfang so klein wie möglich
- Dadurch: höhere Stabilität und Zuverlässigkeit des Systems
- Mikrokernel stellt Kommunikation zwischen Benutzerprogramm und benötigten Betriebssystemdiensten über sog. Message Passing[1] her
- Performance im Vergleich zu Schichtenarchitektur besser, da unnötige Indirektions-Schichten nicht vorhanden
- Allerdings: Monolithische Architektur performanter?

1: Auf das Message Passing wird im Abschnitt Prozessverwaltung näher eingegangen

Hybride Systeme (I)

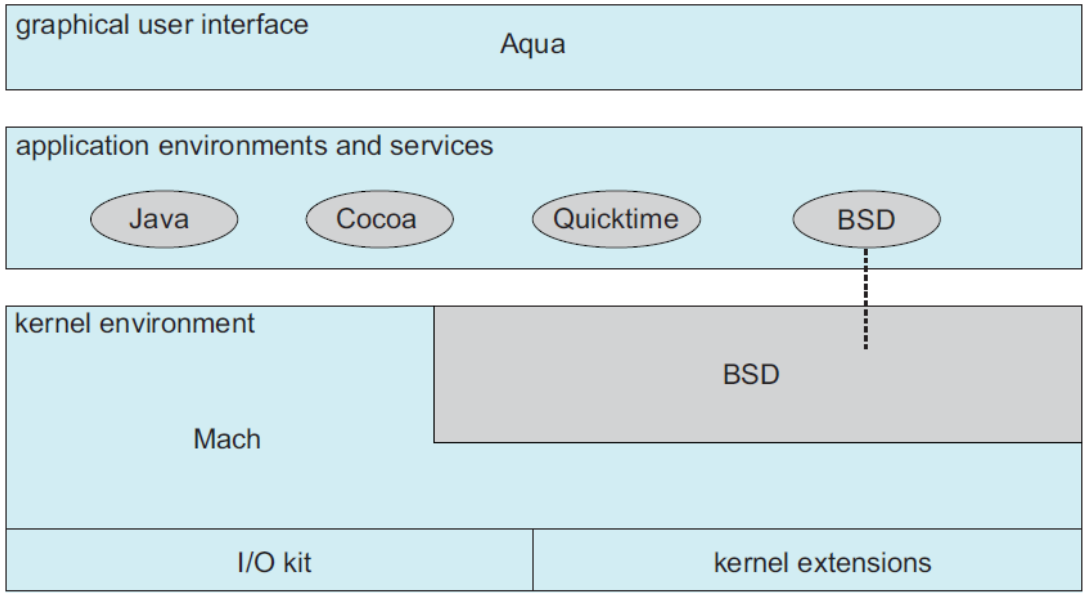
- In der Praxis:

  Moderne Betriebssysteme vereinen zumeist verschiedene Architektur-Ansätze in einem hybriden Gesamtsystem, das zueinander in Konflikt stehende Anforderungen bestmöglich erfüllen soll:

  - Performance
  - Sicherheit
  - Benutzbarkeit
  - Wartbarkeit

- Im Folgenden: Veranschaulichung zweier Beispiel-Architekturen moderner Betriebssysteme

Hybride Systeme (II)

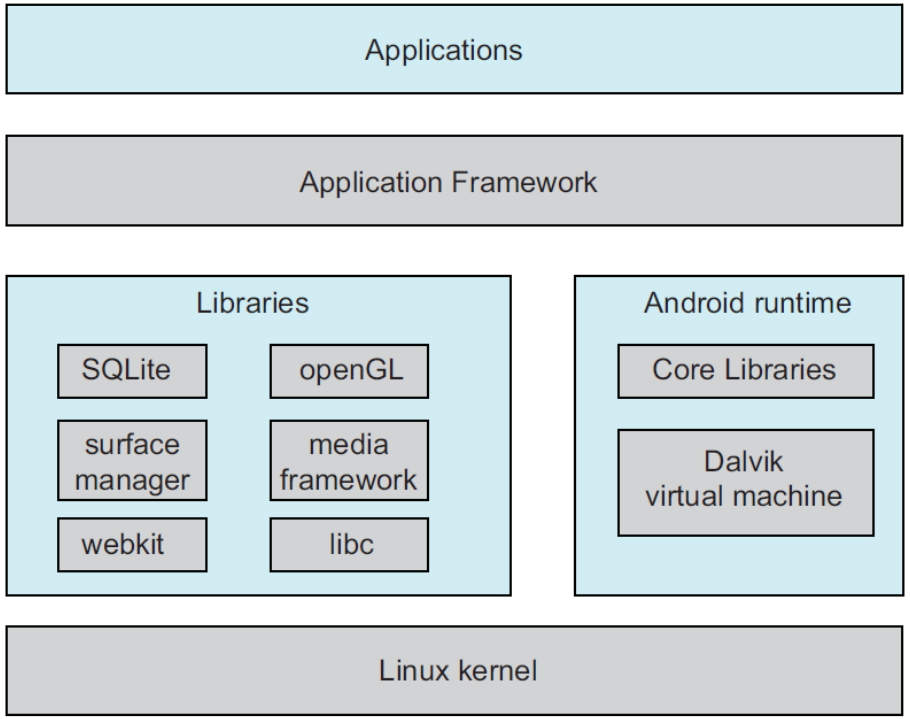


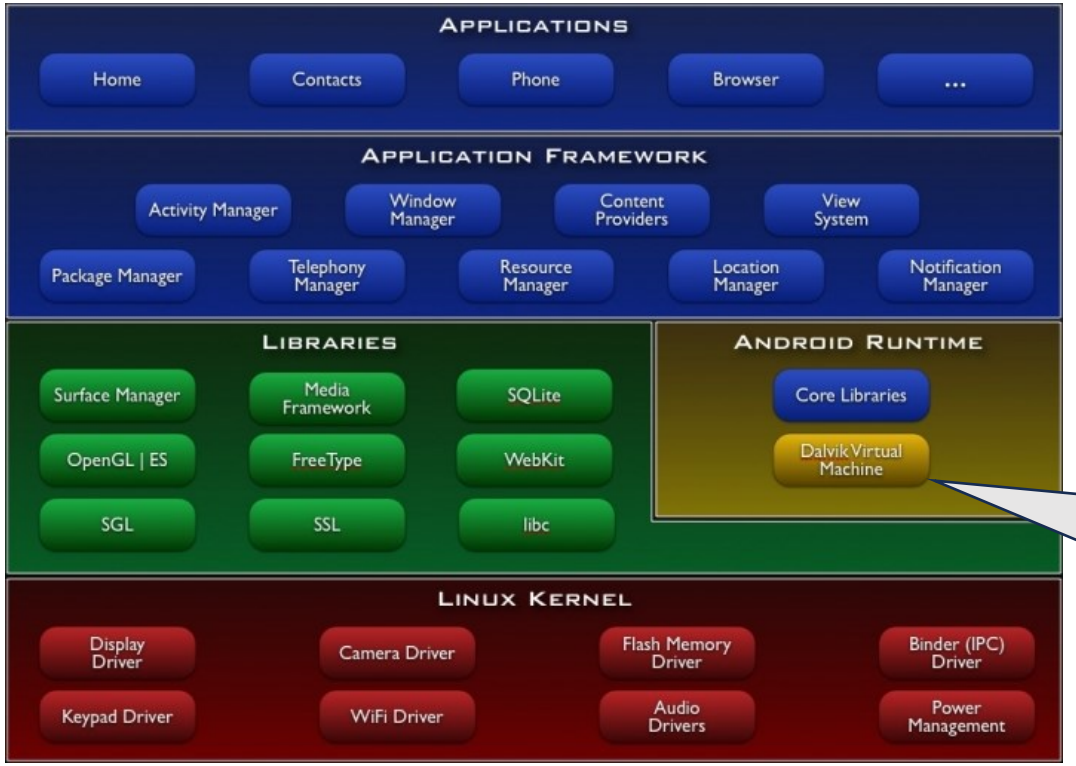
Mac OS X Architektur:

- Schichtenmodell mit Mikrokern
- Cocoa: Programmierschnittstelle für Objective-C
- Mach: Auf Mikrokern-Architektur basierender Betriebssystem-Kernel mit Speicherverwaltung, RPC, IPC, Thread Scheduling, …
- BSD: Command Line Interface, Networking und Dateisysteme, Implementierung der POSIX APIs
- I/O Kit: „SDK“ für Gerätetreiber und dynamisch ladbare Module
- Letztere werden als sogenannte „Kernel extensions“ bezeichnet

Hybride Systeme (III)

- Android-Architektur:





Inzwischen ersetzt durch Android Runtime (ART)

Quelle: *https://elinux.org/Android_Architecture*

Abbildungsverzeichnis

- Alle Abbildungen, sofern nicht anders angegeben aus [MB17] und [AS12]

Literatur

- [BS17] Betriebssysteme – Grundlagen und Konzepte, Rüdiger Brause, 4. Auflage
  Springer Vieweg Verlag, 2017
  ISBN: 978-3-662-54099-2

- [GB14] Grundkurs Betriebssysteme, Peter Mandl, 4. Auflage
  Springer Vieweg Verlag, 2014
  ISBN: 978-3-658-06217-0

- [BK17] Betriebssysteme Kompakt, Christian Baun, 1. Auflage
  Springer Vieweg Verlag, 2017
  ISBN: 978-3-662-53142-6

Literatur

- [MB17] Moderne Betriebssysteme, Andrew S. Tanenbaum & Herbert Bos, 4. Auflage
  Pearson Studium, 2017
  ISBN: 978-3-86894-270-5

- [MS12] Multicore-Software, Urs Gleim & Tobias Schüle
  dpunkt.verlag, 2012
  ISBN: 978-3-89864-758-8

- [AS12] Operating System Concepts, Silberschatz et al.
  John Wiley & Sons Inc; Revised Edition, 2012
  ISBN: 978-1-11806-333-0