Systemnahe Software
WS 2006/2007

Andreas F. Borchert
Universität Ulm

22. Januar 2007

Betriebssysteme

Die *DIN-Norm 44300* definiert ein Betriebssystem wie folgt:

> *Zum Betriebssystem zählen die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die Basis der möglichen Betriebsarten des digitalen Rechensystems bilden und die insbesondere die Abwicklung von Programmen steuern und überwachen.*

Start des Betriebssystems

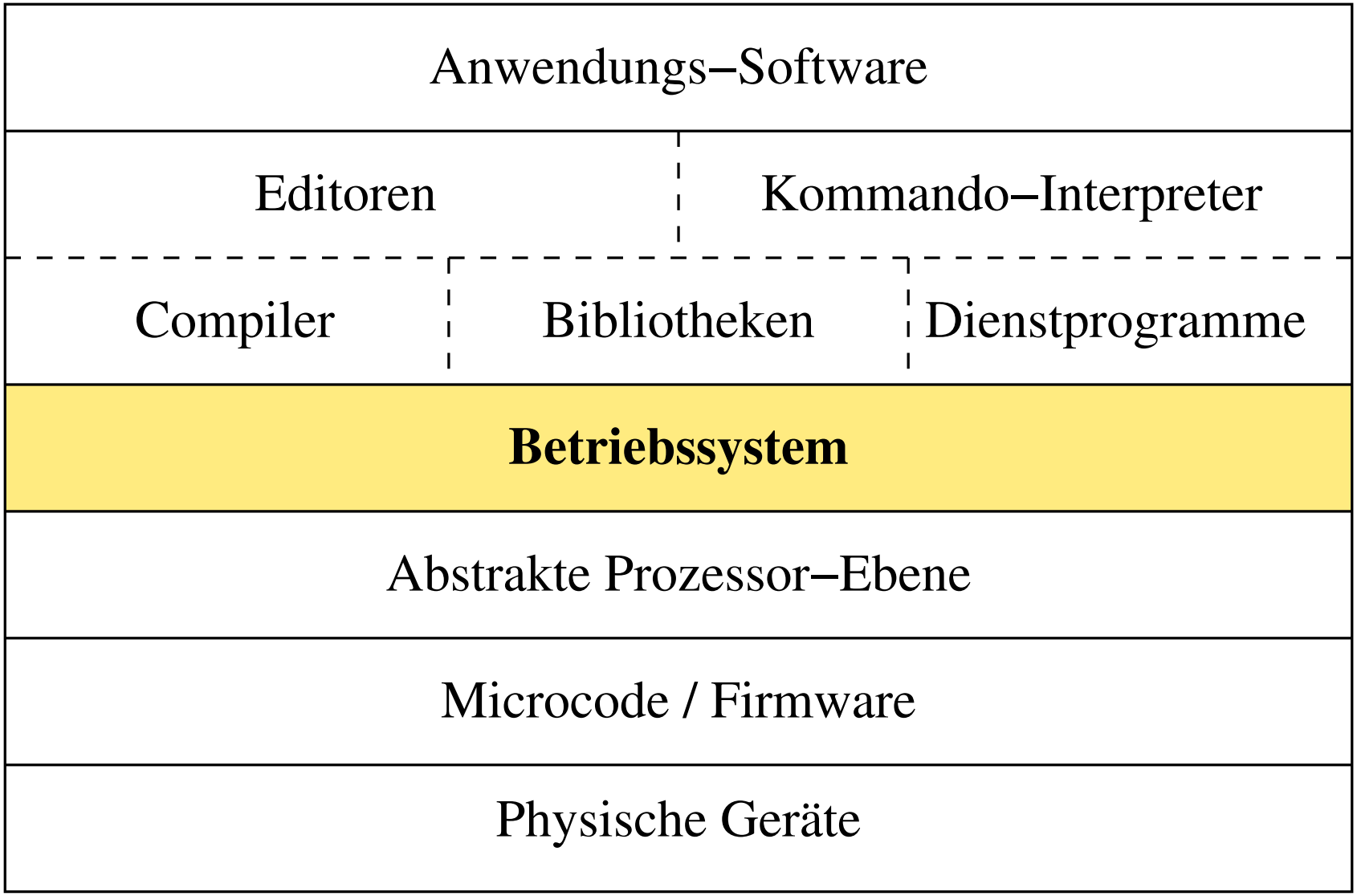
- Das Betriebssystem ist (abgesehen von der Firmware und einigen Zwischenstufen) das erste Programm, das von einem Rechner beim Hochfahren geladen wird.
- Das Betriebsystem läuft die gesamte Zeit, bis der Rechner wieder heruntergefahren wird.

Aufgaben des Betriebssystems

Das Betriebssystem hat zwei zentrale *Aufgaben*:

- **Ressourcen-Management:** Das Betriebssystem verwaltet und kontrolliert alle Hardware- und Software-Komponenten eines Rechners und teilt sie möglichst fair und effizient den einzelnen Nachfragern zu.
- **Erweiterte oder virtuelle Maschine:** Das Betriebssystem besteht aus einer (oder mehreren) Software-Schichten, die über der „nackten“ Hardware liegen. Diese erweiterte Maschine ist einfacher zu verstehen und zu programmieren, da sich komplizierte Zugriffe und Abhängigkeiten hinter einer einfacheren und einheitlichen Schnittstelle verbergen – den Systemaufrufen.

Schichtenmodell



Schichtenmodell

- **Physische Geräte:**
  Prozessor, Festplatten, Grafikkarte, Stromversorgung, etc.
- **Microcode / Firmware:**
  Software, die die physikalischen Geräte direkt kontrolliert und sich teilweise direkt auf den Geräten befindet. Diese bietet der nächsten Schicht eine einheitlichere Schnittstelle zu den physikalischen Geräten. Dabei werden einige Details der direkten Gerätesteuerung verborgen.

  Beispiel: Abbildung logischer Adressen auf physische Adressen bei Festplatten.
- **Abstrakte Prozessor-Ebene:**
  Schnittstelle zwischen Hard- und Software. Hierzu gehören nicht nur alle Instruktionen des Prozessors, sondern auch die Kommunikationsmöglichkeiten mit den Geräten und die Behandlung von Unterbrechungen.

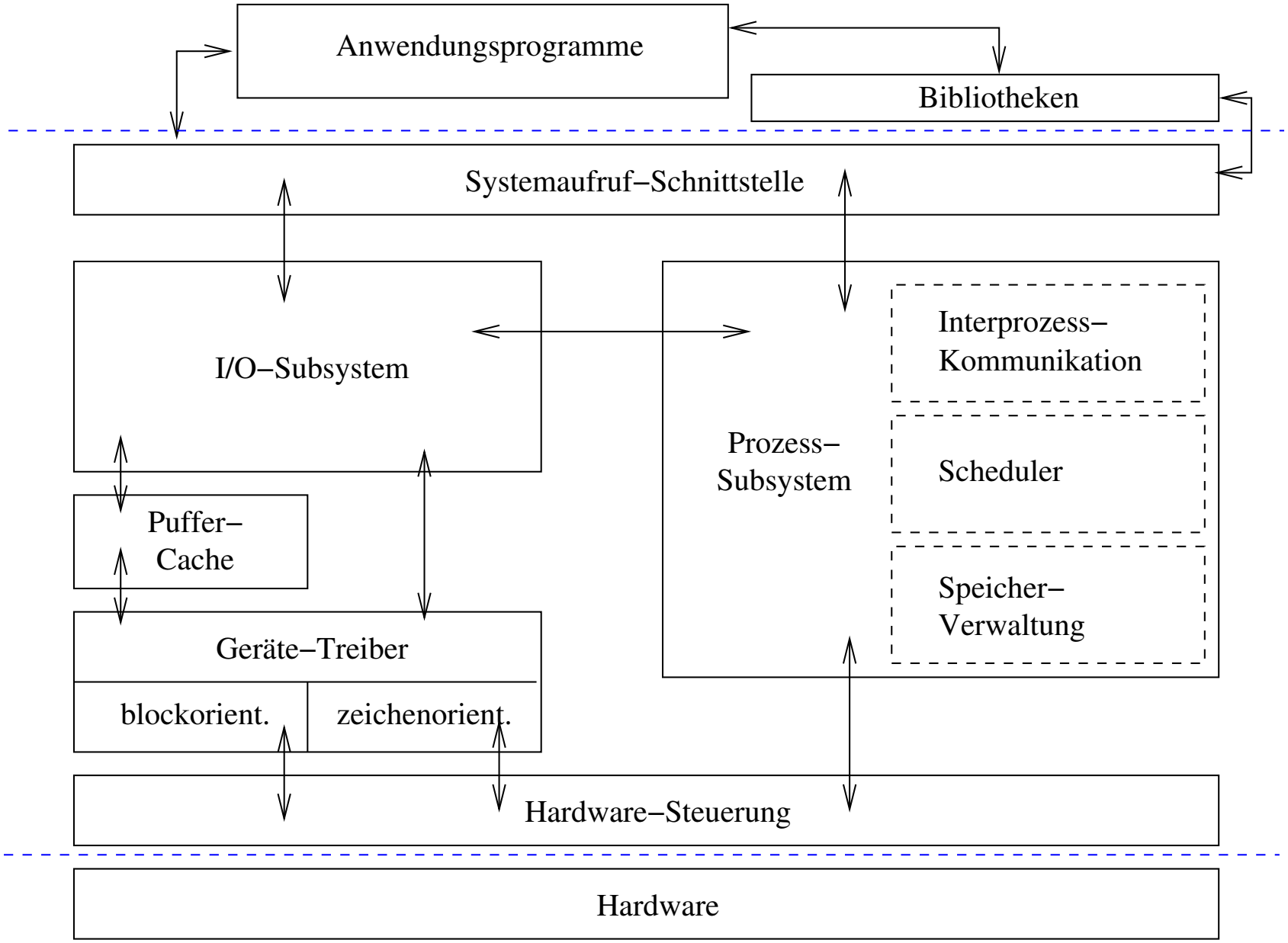
Schichtenmodell

- **System-Software:**
  Software, die von der Schnittstelle des Betriebssystems abhängt und typischerweise vom Hersteller des Betriebssystems mit ausgeliefert wird.

  Beispiele: Bibliotheken (libc.a), Kommandozeilen-Interpreter (Shells), graphische Benutzeroberflächen (X-Windows), systemnahe Werkzeuge, Netzwerkdienste (Web-Server)

- **Anwendungen:**
  Von Benutzern bzw. für Benutzer zur Lösung ihrer Probleme entwickelte Programme *Beispiel:* Textverarbeitungsprogramm

Interner Aufbau von Unix

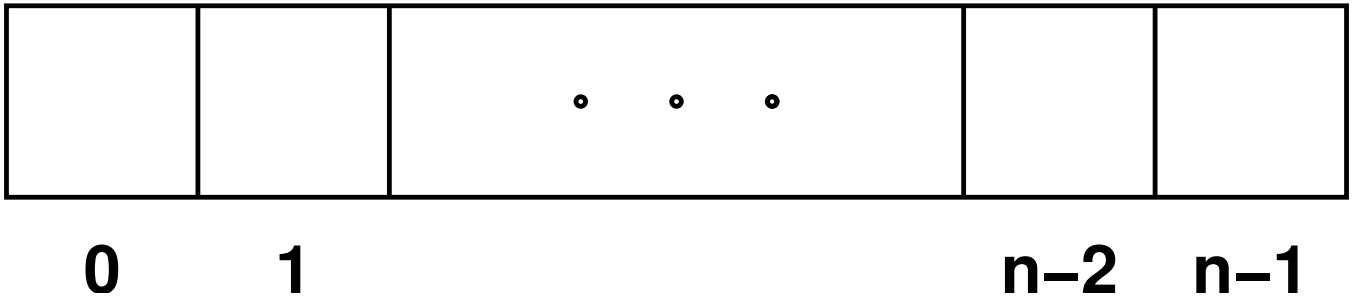


Definition einer Datei

Aus IEEE Std 1003.1 (POSIX):

> *An object that can be written to, or read from, or both. A file has certain attributes, including access permissions and type. File types include regular file, character special file, block special file, FIFO special file, symbolic link, socket, and directory. Other types of files may be supported by the implementation.*

Modell einer gewöhnlichen Datei



- Eine gewöhnliche Datei entspricht einem Array aus Bytes.
- Wenn eine Datei eine Länge von n Bytes hat, sind diese über die Positionen 0 bis $n - 1$ abrufbar.
- Eine Dateiverbindung hat eine aktuelle Position p.
- Wenn ein Byte über eine Verbindung gelesen oder geschrieben wird, dann erfolgt der Zugriff auf der aktuellen Position p, die anschließend, falls die Operation erfolgreich war, um eins erhöht wird.
- Lese-Operationen bei einer Position von n sind nicht erfolgreich.

Struktur einer Datei

- Unix verlangt und unterstellt bei regulären Dateien *keinerlei Struktur* und unterstützt auch keine.
- Die Konzepte variabel oder konstant langer Datensätze (Records) sind im Kernel von UNIX nicht implementiert.
- Entsprechend sind gewöhnliche Dateien ganz schlichte Byte-Arrays.
- Die einzige Besonderheit ist, dass Dateien unter Unix „Löcher“ haben dürfen, d.h. einzelne Indexbereiche des Arrays können unbelegt sein. Diese werden dann als Nullbytes ausgelesen.

Verwaltungsinformationen einer Datei

Zu einer Datei gehören

- ein oder auch mehrere *Namen*,
- der *Inhalt* und *Aufbewahrungsort* (Menge von Blöcken auf der Platte, etc.) und
- *Verwaltungsinformationen* (Besitzer, erlaubter Zugriff, Zeitstempel, Länge, Dateityp, etc.).

Spezielle Dateien

- Neben den gewöhnlichen Dateien gibt es unter Unix weitere Dateiformen.
- Neben den Verzeichnissen gibt es insbesondere Dateivarianten, die der Interprozess-Kommunikation oder direkten Schnittstelle zu Treibern des Betriebssystems dienen.
- Diese weichen in der Semantik von dem Byte-Array ab und bieten beispielsweise uni- oder bidirektionale Kommunikationskanäle.

Gerätedateien

- Gerätedateien erlauben die direkte Kommunikation mit einem (das jeweilige Gerät repräsentierenden) Treiber.
- Sie erlauben beispielsweise den direkten Zugriff auf eine Festplatte vorbei an dem Dateisystem.
- Für Gerätedateien gibt es zwei verschiedene Schnittstellen:
  - **Zeichenweise arbeitende Geräte** (*character devices / raw devices*):
    Diese Dateien erlauben einen ungepufferten zeichenweisen Lese- und/oder Schreibzugriff.
  - **Blockweise arbeitende Geräte** (*block devices*):
    Diese Dateien erlauben Lese- und Schreibzugriffe nur für vollständige Blöcke. Diese Zugriffe laufen implizit über den Puffer-Cache von Unix.

Zugriffe auf eine Platte

Auf eine Festplatte kann typischerweise auf drei verschiedene Weisen zugegriffen werden:

- Über ein Dateisystem.
- Über die zugehörige blockweise arbeitende Gerätedatei indirekt über den Puffer-Cache.
- Über die zugehörige zeichenweise arbeitende Gerätedatei.

Intern im Betriebsystem liegt die gleiche Schichtenstruktur der Schnittstellen vor: Zugriffe auf ein Dateisystem werden abgebildet auf Zugriffe auf einzelne Blöcke innerhalb des Puffer-Cache. Wenn der gewünschte Block zum Lesen nicht vorliegt oder ein verändeter Block im Cache zu schreiben ist, dann wird der zugehörige Treiber direkt kontaktiert.

Arten von Dateisystemen

Prinzipiell lassen sich Dateisysteme in vier Gruppen unterteilen:

- *Plattenbasierte Dateisysteme:*
  Die Daten des Dateisystems liegen auf einer lokalen Platte.
- *Netzwerk-Dateisystem:*
  Das Dateisystem wird von einem anderen Rechner über das Netzwerk angeboten. Beispiele: NFS, AFS und Samba.
- *Meta-Dateisysteme:*
  Das Dateisystem ist eine Abbildungsvorschrift eines oder mehrerer anderer Dateisysteme. Beispiele: *tfs* und *unionfs*.
- *Pseudo-Dateisystem:*
  Das Dateisystem ist nicht mit persistenten Daten verbunden. Beispiel: Das *procfs* unter */proc*, das die einzelnen aktuell laufenden Prozesse repräsentiert.

Plattenbasierte Dateisysteme

- Gegeben ist die abstrakte Schnittstelle eines Arrays von Blöcken. (Dies kann eine vollständige Platte sein, eine Partition davon oder eine virtuelle Platte, wie sie etwa bei diversen RAID-Verfahren entsteht.)
- Zu den Aufgaben eines plattenbasierten Dateisystems gehört es, ein Array von Blöcken so zu verwalten, dass
  - über ein hierarchisches Namenssystem
  - Dateien (bis zu irgendeinem Maxium) frei wählbarer Länge
  - gespeichert und gelesen werden können.

Integrität eines Dateisystems

Aus dem Werk von Marc J. Rochkind, Seite 29, zum Umgang mit einer Schreib-Operation:

> *I've taken note of your request, and rest assured that your file descriptor is OK,*
>
> *I've copied your data successfully, and there's enough disk space. Later, when it's convenient for me, and if I'm still alive, I'll put your data on the disk where it belongs.*
>
> *If I discover an error then I'll try to print something on the console, but I won't tell you about it (indeed, you may have terminated by then).*
>
> *If you, or any other process, tries to read this data before I've written it out, I'll give it to you from the buffer cache, so, if all goes well, you'll never be able to find out when and if I've completed your request.*
>
> *You may ask no further questions. Trust me. And thank me for the speedy reply.*

Integrität eines Dateisystems

Was passiert, wenn dann mittendrin der Strom ausfällt?

- Blöcke einer Datei oder gar ein Verwaltungsblock sind nur teilweise beschrieben.
- Verwaltungsinformationen stimmen nicht mit den Dateiinhalten überein.

(Wieder-)herstellung der Integrität

Im Laufe der Zeit gab es mehrere Entwicklungsstufen bei Dateisystemen in Bezug auf die Integrität:

- Im Falle eines Falles muss die Integrität mit speziellen Werkzeugen überprüft bzw. hergestellt werden. Beispiele: Alte Unix-Dateisysteme wie UFS (alt), ext2 oder aus der Windows-Welt die Familie der FAT-Dateisysteme.
- Ein Journalling erlaubt normalerweise die Rückkehr zu einem konsistenten Zustand. Beispiele: Neuere Versionen von UFS, ext3 und reiser3.
- Das Dateisystem ist immer im konsistenten Zustand und arbeitet entsprechend mit Transaktionen analog wie Datenbanken. Hinzu kommen Überprüfungssummen und Selbstheilungsmechanismen (bei redundanten RAID-Verfahren). Beispiele: ZFS, ext4, reiser4 und NTFS.