Betriebssysteme

Kapitel 5 Ein- und Ausgabe

Ziele der Vorlesung

- Einführung
  - -Historischer Überblick
  - -Betriebssystemkonzepte
- Prozesse und Threads
  - -Einführung in das Konzept der Prozesse
  - -Prozesskommunikation
  - -Scheduling von Prozessen
  - -Threads
- Speicherverwaltung
  - -Einfache Speicherverwaltung
  - -Virtueller Speicher
  - -Segmentierter Speicher
- Dateien und Dateisysteme
  - -Dateien
  - -Verzeichnisse
  - -Implementierung von Dateisystemen
- Grundlegende Eigenschaften der I/O-Hardware
  - -Festplatten
  - -Terminals
  - -Die I/O-Software
- Deadlocks/Verklemmungen
- Virtualisierung und die Cloud
- Multiprozessor-Systeme
- IT-Sicherheit
- Fallstudien

Ein- und Ausgabe

- Hardware-Geräte zur Ein- und Ausgabe (Sicht Elektrotechniker)
  - Physische Komponenten der Ein-/Ausgabegeräte (Chips, Drähte, Motoren, Stromversorgung,…)

- **Software zur Ein- und Ausgabe** (Sicht Programmierer)
  - Kommandos, die die Hardware kennt,
  - Funktionen, die die Hardware ausführt
  - Mögliche Fehlermeldungen

- Programmierung von Ein-/Ausgabengeräten ist oft eng mit deren internem Aufbau verbunden

Hardware zur Ein- und Ausgabe

- Zwei Klassen
  - **Blockorientierte Geräte** (block devices)
    - Informationen werden in Blöcken fester Größe abgespeichert
    - Jeder Block besitzt eine eigene Adresse
    - Blockgrößen in der Regel von 512 Byte bis 65536 Byte
    - Jede Übertragung läuft in Einheiten von einem oder mehreren ganzen (aufeinadnerfolgenden) Blöcken ab
    - Jeder Block kann unabhängig von anderen Blöcken gelesen oder geschrieben werden

      → Bsp: Festplatte, USB-Stick, CD-ROM, …

  - **Zeichenorientierte Geräte** (character devices)
    - Zeichenströme werden erzeugt oder akzeptiert
      - ohne interne Blockstruktur
    - Nicht adressierbar und kennt keine Suchfunktion

      → Bsp: Tastatur, Maus, Netzwerkkarte, …

Typische Datenraten von E/A-Geräten, Netzwerken und Bussystem

Gerät	Datenrate
Tastatur	10 Byte/s
Maus	100 Byte/s
56-KB-Modem	7 KB/s
Scanner	400 KB/s
Digitaler Camcorder	3,5 MB/s
WLAN nach 802.11g	6,75 MB/s
52-fach CD-ROM	7,8 MB/s
Fast Ethernet	12,5 MB/s
Compact Flash Card	40 MB/s
FireWire (IEEE 1394)	50 MB/s
USB 2.0	60 MB/s
SONET-OC-12-Netwerk	78 MB/s
Ultra-2-SCSI-Platte	80 MB/s
Gigabit Ethernet	125 MB/s
SATA-Plattenlaufwerk	300 MB/s
Ultrium-Bandlaufwerk	320 MB/s
PCI-Bus	528 MB/s

Hardware zur Ein- und Ausgabe

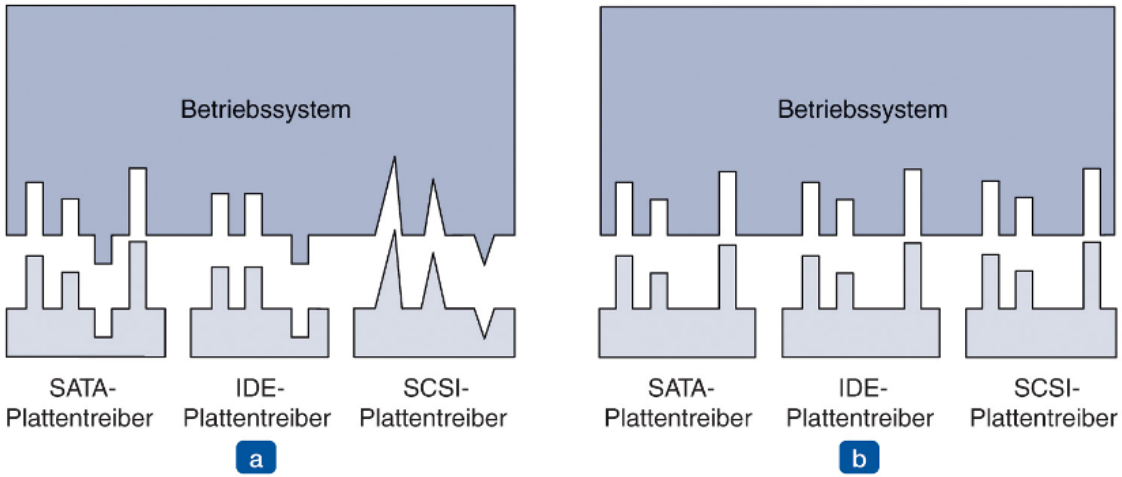
Vernetzungstechniken

Technik	Reichweite im Gebäude	Datenrate auf Medium	typische Datenrate auf Anwendungsebene
TV-Kabel (Koax)	bis 600 Meter Kabellänge	max. 200/500 MBit/s	30 bis 200 MBit/s
Telefonkabel (Zweidrahtleitung)	bis 600 Meter Kabellänge	max. 200/500 MBit/s	80/200 MBit/s
Powerline (HomePlug AV)	bis 200 Meter Kabellänge	max. 200/500 MBit/s	20 bis 200 MBit/s
Bluetooth 3.0	typisch 20 Meter	max. 3 MBit/s (54 MBit/s[1])	0,9 bis 2,7 MBit/s (5 bis 20 MBit/s[1])
WLAN IEEE 802.11g	typisch 20 Meter	max. 54 MBit/s	5 bis 20 MBit/s
WLAN IEEE 802.11n	typisch 20 Meter	max. 72 / 144 / 300 / 450 / 600 MBit/s[2]	40 bis 200 MBit/s
WLAN IEEE 802.11ac	typisch 20 Meter	max. 87 / 180 / 390 / 867 / 1333 ... 6933 MBit/s [2]	50 bis 400 MBit/s
WLAN IEEE 802.11ad	typisch 5 Meter (Zimmer)	max. 4620 / 6757 MBit/s	noch unbekannt [3]
Ethernet (Fast/Gigabit /10GBaseT)	bis 100 Meter Kabellänge	100/1000/10 000 MBit/s	93/930/9300 MBit/s

[1] mit WLAN als sekundärem Medium [2] abhängig von Implementierung und Konfiguration (1 bis 8 Antennen, 20 bis 160 MHz Kanalbreite) [3] erste Geräte erscheinen voraussichtlich noch in 2013

https://silo.tips/download/netzwerke-netzwerke-nas-statt-server-komplettpaket-test-und-technik-software-fr

Ziel: Einheitliches System für Ein- und Ausgabe



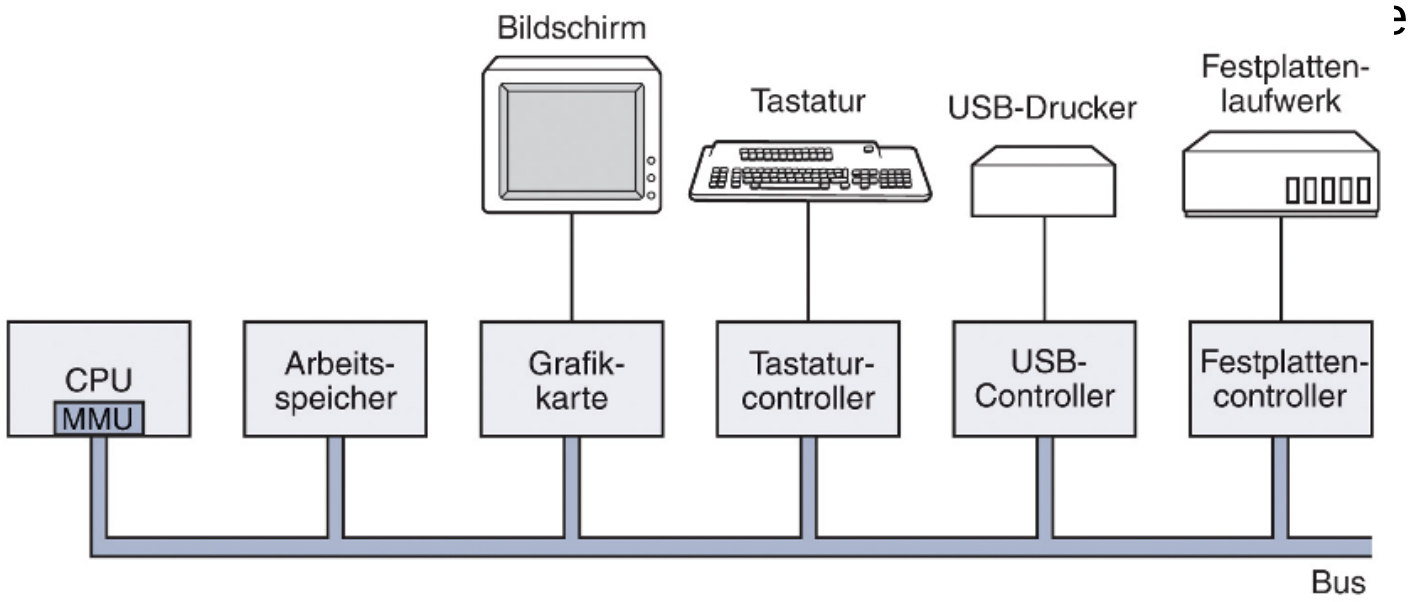
(a) Ohne eine Standardschnittstelle für Treiber; (b) Mit einer Standardschnittstelle für Treiber

(a) Jeder Gerätetreiber hat eine andere Schnittstelle zum Betriebssytsem. Die Treiberfunktionen, die dem System zum Aufruf zur Verfügung stehen, sind von Treiber zu Treiber unterschiedlich. Das Einbinden jedes neuen Treibers führt zu einer Menge Programmieraufwand.

(b) Einheitliche Schnittstelle für alle Geräte. Es ist einfach einen neuen Treiber einzubinden, vorausgesetzt er hält sich an die gemeinsame Schnittstelle.
In der Praxis sind zwar nicht alle Geräte absolut identisch, aber i. A. gibt es nur eine kleine Anzahl von Geräteklassen.Oft sind diese meistens weitgehend identisch.

Ziel: Einheitliches System für Ein- und Ausgabe

- Ein-/Ausgabe-Einheiten bestehen häufig aus
  - einer mechanischen und
  - einer elektronischen Komponente
- **Mechanische Komponente = Gerät**
- **Elektronische Komponente = Controller, Gerätesteuereinheit, Adapter**



Hardware zur Ein- und Ausgabe

- Controller
  - Controllerkarte hat meist Steckverbindung um Kabel mit dem Gerät zu verbinden
  - Controller können mehrere identische Geräte verwalten
  - Spezielle Hardware, oft mit eigenem Mikroprozessor
    - Besitzt einige Register für Kommunikation
      - Betriebssystem schreibt in die Register, um Befehle zu erteilen
      - Betriebssystem erhält Informationen über das Gerät
  - Besitzt evtl. einen Datenpuffer
    - Betriebssystem kann lesen und schreiben
  - Steuert das Gerät weitgehend autonom
    - Kann interrupts melden
  - Bietet vereinfachte (aber noch komplexe) Schnittstelle

Hardware zur Ein- und Ausgabe

- Controller
  - Schnittstelle zwischen Controller und Gerät ist oft standardisiert (offizieller oder De-Facto-Standard)
    - Offizieller Standard: ANSI, IEEE, ISO,….
    - De-Facto-Standard: SATA-, SCSI-, USB-Schnittstellen
  - Firmen können Geräte und Controller produzieren, die auf diesen Schnittstellen verwenden
  - Schnittstelle oft auf maschinennaher Ebene
    - Beispiel:
      - Festplatte mit 2 Mill. Sektoren, 512 Byte formatiert
      - Festplatte liefert seriellen Bitstrom, der mit einer Präambel beginnt, von 4096 Bit des Sektors gefolgt wird und am Ende noch eine Prüfsumme oder fehlerkorrigierenden Code (ECC=Error-Correcting Code) enthält.
      - Präambel wird festgelegt bei Formatierung (enthält Zylinder- und Sektornummer, Sektorgröße, Synchinfos usw.
  - Controller konvertiert seriellen Bitstrom in Byteblöcke und führt Fehlerkorrekturen durch
  - Block wird in Puffer des Controllers Bit für Bit gesammelt
  - Prüfsumme OK (=Block fehlerfrei)→ Kopieren in Arbeitsspeicher

Aufgabe/Frage

Wie muss man sich die Arbeit des Controllers für einen LCD-Monitor vorstellen?

Bedingung:

→ 3 min

Lösung

Antwort:

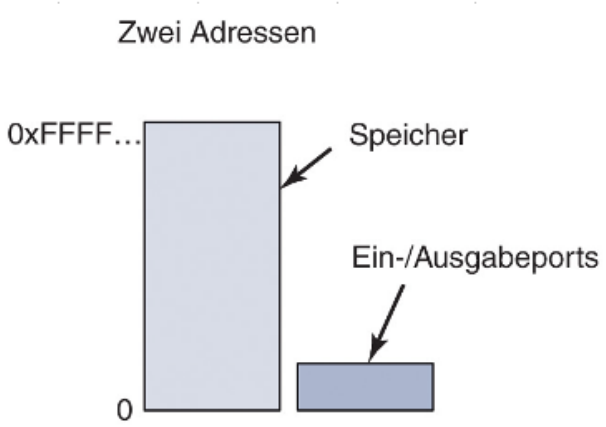
- Controller arbeitet auch als bitserielles Gerät
  - liest Daten aus dem Arbeitsspeicher, der die darzustellende Zeichen enthält
  - Generiert Signale, um die Polarisierung der Hintergrundbeleuchtung für die entsprechenden Pixel zu modifizieren und auf den Bildschirm zu schreiben
- Betriebssystem initialisiert den Controller mit Parametern (Anzahl Zeichen pro Zeile, Anzahl der Zeilen pro Bildschirmseite)
- Controller steuert die elektrischen Felder

Hardware zur Ein- und Ausgabe

- Controller
  - Controller besitzt **einige Register** für die Kommunikation mit dem Prozessor
    - Durch das Schreiben in diese Register kann Betriebssystem dem Gerät Befehle erteilen
    - Durch das Lesen dieser Register erhält das Betriebssystem Infos über das Gerät (z.B. aktueller Zustand, Bereitschaft Befehle zu bearbeiten)
  - Zusätzlich zu Kontrollregister besitzen viele Geräte auch **Datenpuffer**, die das Betriebssystem lesen und schreiben kann (z.B. Videospeicher für die Darstellung von Pixelns auf dem Bildschirm)
  - Es gibt 2 Alternativen wie Betriebssystem mit Kontrollregistern und Datenpuffern kommuniziert.
    - **Eingabe-/Ausgabe-Port-Nummern (I/O Port Number)** werden jedem Kontrollregister zugewiesen (eine 8-Bit oder 16-Bit-Zahl). Die Menge aller Ein-/Ausgabeports bildet den Ein-Ausgabeport-Namensraum (geschützt)
    - **Speicherbasierte Ein-/Ausgabe (memory mapped I/O),** alle Kontrollregister werden in den Arbeitsspeicher eingeblendet

Hardware zur Ein- und Ausgabe

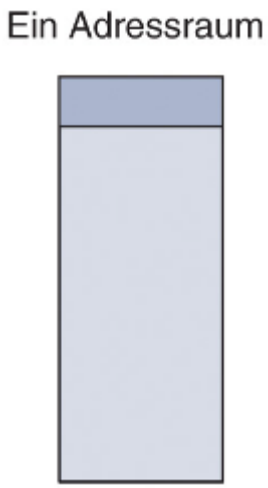
- Kommunikation Controller <--> Betriebssystem
  - **Über Eingabe/Ausgabeport-Nummern (I/O port number)**
    - Jedes Kontrollregister erhält eine E/A-Port-Nummer
    - Nur durch Betriebssystem zugreifbar mit Hilfe spezieller Befehle
    - Bsp.:
      IN REG, PORT (Prozessor liest das Kontrollregister PORT in ein Register REG)
      OUT PORT, REG (schreibt den Inhalt von Register REG in PORT)
    - Unterschiedlicher Adressraum für Arbeitsspeicher und E/A-Geräte (fast alle Großrechner)



Getrennter Ein-/Ausgabe- und Arbeitsspeicherbereich

Hardware zur Ein- und Ausgabe

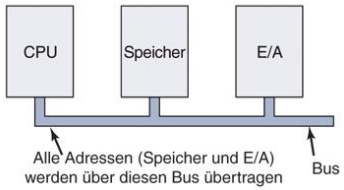
- Kommunikation Controller <--> Betriebssystem
  - **Speicherbasierte E/A (memory mapped I/O)**
    - Einblenden der Controller-Register in den Adressraum
      - Jedes Controller-Register erhält eindeutige Speicheradresse zugewiesen, zu der kein Arbeitsspeicher vorhanden ist
      - Zugewiesene Adressen liegen meist im oberen Adressraum
      - **Zugriff auf die Geräte wie bei Zugriff auf Arbeitsspeicher**
      - **Vorteile:**
        - Gleiche Routine wie bei Speicherzugriff; jeder Befehl, der Speicheradressen ansprechen kann, kann auch Kontrollregister adressieren
        - Weniger Assembler Code als bei E/A-Port-Nummern
        - Kein spezieller Schutzmechanismus notwendig, der Benutzer-programme vom Zugriff auf E/A-Geräte abhält
          - → Betriebssystem darf keine Teile des Adressraums mit den Kontrollregistern in den virt. Adressraum von Benutzerprozessen einblenden
          - → Anzahl der Zugriffe auf Register kann reduziert werden, wenn verschiedene Gerätetreiber in unterschiedlichen Adressräumen untergebracht werden



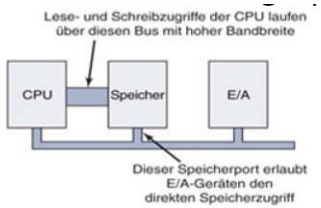
E/A-Adressraum liegt auch im Arbeitsspeicher

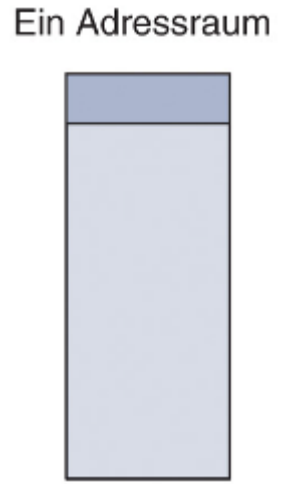
Hardware zur Ein- und Ausgabe

- Kommunikation Controller <--> Betriebssystem
  - **Speicherbasierte E/A (memory mapped I/O)**
    - Einblenden der Controller-Register in den Adressraum
      - **Nachteile:**
        - Rechner können Speicherwörter zwischenspeichern; das Cachen eines Kontrollregisters wäre jedoch fatal
          → Hardware muss selektives Ausschalten von Caching unterstützen, Betriebssystem muss es verwalten
        - Bei nur einem Adressraum müssen alle Speichermodule und alle Ein-/Ausgabegeräte jeden einzelnen Speicherzugriff untersuchen, um festzustellen, wer mit der Adresse angesprochen wird.
          - Einfach bei einem Bus



          - PCs haben eigenen Hochgeschwindigkeitsbus für Speicherzugriff; kein Kompromiss für langsame E/A-Geräte





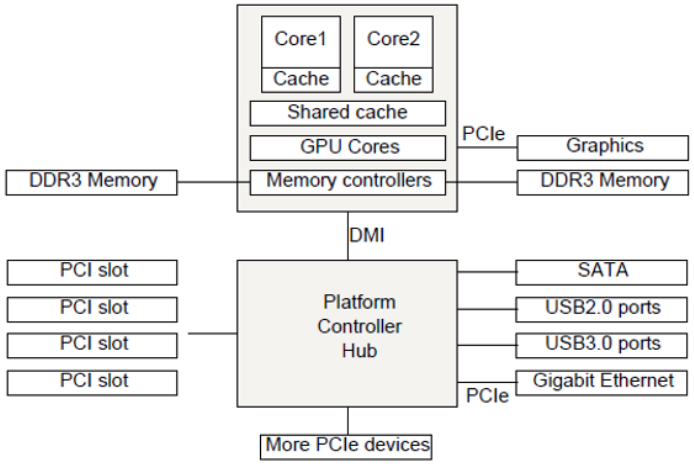
E/A-Adressraum liegt auch im Arbeitsspeicher

Hardware zur Ein- und Ausgabe

- Kommunikation Controller <--> Betriebssystem
  - **Speicherbasierte E/A (memory mapped I/O)**
    - Problem bei getrennten Speicherbussen: die Ein-/Ausgabegeräte können die Adressen auf dem Speicherbus nicht sehen und deshalb nicht reagieren
    - Lösungen:
      - Alle Speicherzugriffe werden zunächst zum Speicher geschickt. Antwortet der Speicher nicht, versucht es der Prozessor auf den anderen Bussen.
      - Installation eines eigenen Geräts an den Speicherbus, das alle Adressen mithört und an entsprechende Geräte weiterleitet. Das Problem dabei ist, daß die Geräte nicht mehr mit derselben Geschwindigkeit reagieren als der originäre Speicher.
      - Filtern von Adressen im Speichercontroller

Hardware zur Ein- und Ausgabe

- Kommunikation Controller <--> Betriebssystem
  - **Speicherbasierte E/A (memory mapped I/O)**
    - Lösungen
      - ➢ Filtern von Adressen im Speichercontroller



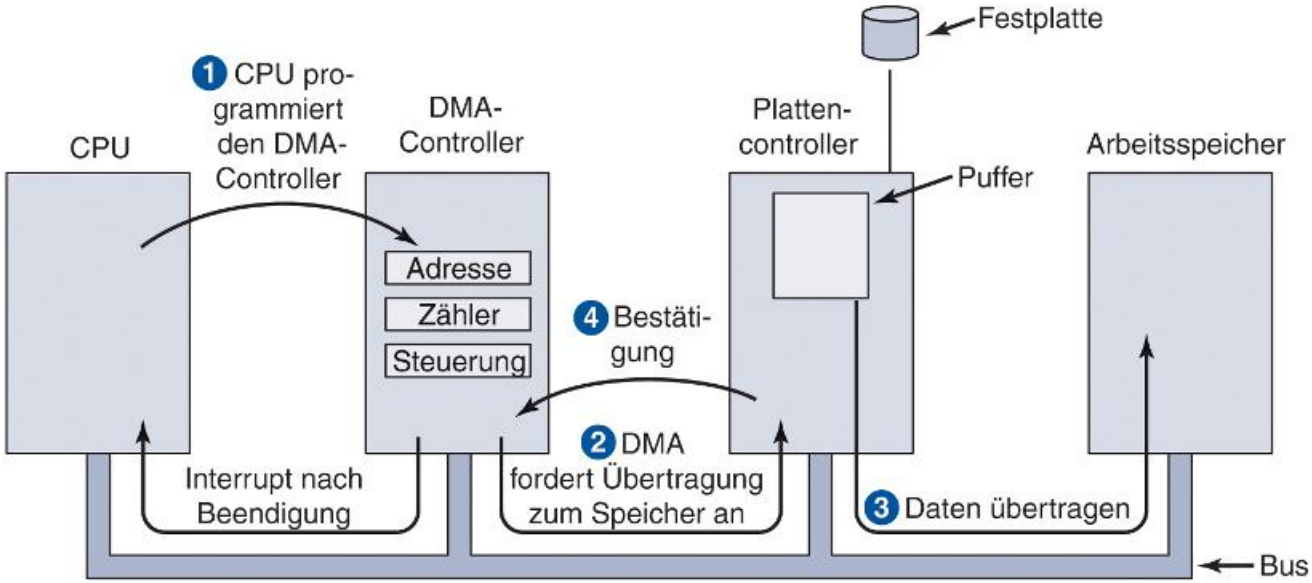
Aufbau eines ausgebauten x86 Systems

- ➢ Der Speichercontrollerchip enthält eine Reihe von Registern, die beim Hochfahren geladen werden. z.B. Markierung von 640KB bis 1 MB-1 ohne Speicher.
- ➢ Weiterleitung aller Adressen aus diesem Bereich werden zu Geräten weitergeleitet anstatt sie auf den Speicherbus zu legen.
- ➢ **Nachteil:** Es muss zur Startzeit des Systems ermittelt werden, welche Adressen keine echten Speicheradressen sind.

PCIe=Peripheral Component Interconnect Express
SCSI=Small Computer System Interface
SATA=Serial AT Attachment
USB=Universal Serial Bus
DMI=Direct Media Interface

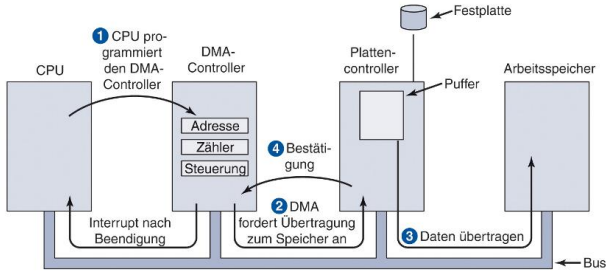
Hardware zur Ein- und Ausgabe

- Kommunikation Controller <--> Betriebssystem
  - Bisher: CPU ist für das Holen der Daten vom Controller verantwortlich
  - Zur Vereinfachung: CPU greift auf alle Geräte auch den Speicher über ein einziges Bussystem zu, das die CPU, den Speicher und die Ein-/Ausgabegeräte verbindet.



Hardware zur Ein- und Ausgabe

- Kommunikation Controller <--> Betriebssystem
  - **Direct Memory Access (DMA)** ist anderes Verfahren
    - Hardware muss DMA Controller haben
    - DMA Controller ist entweder in einen Plattencontroller oder die Steuereinheit anderer Geräte integriert (jeweils ein eigenständiger DMA-Controller pro Gerät) ODER
      DMA-Controller ist als einziger DMA-Steuerbaustein auf der Hauptplatine um den Datentransfer zu regeln.
    - DMA-Controller hat immer Zugriff auf Systembus
    - DMA-Controller enthält mehrere Register
      - Speicheradressregister,
      - Bytezählregister und
      - Ein oder mehrere Kontrollregister
        - bestimmen den Ein-/Ausgabeport
        - die Richtung der Datenübertragung (Lesen oder Schreiben auf das Gerät)
        - die Übertragungseinheit (Byte oder Wort)
        - die Anzahl der Daten, die in einem Zyklus übertragen werden

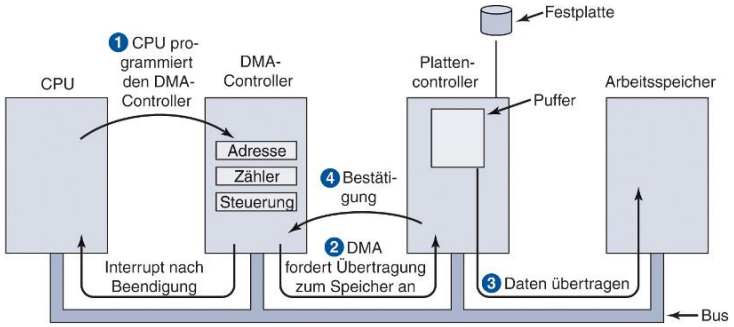


Hardware zur Ein- und Ausgabe

- Kommunikation Controller <--> Betriebssystem
  - **Direct Memory Access (DMA)**
    - **Lesen von Festplatte ohne DMA**
      - Plattencontroller liest den Block Byte für Byte von Platte, bis der gesamte Block im internen Puffer des Controllers liegt.
      - Berechnung einer Prüfsumme (keine Lesefehler aufgetreten)
      - Controller erzeugt Interrupt
      - Sobald Betriebssystem Rechenzeit bekommt, wird der Plattenblock aus dem Speicher des Controllers gelesen
      - Schleife liest Zeichen oder Wort aus dem Geräteregister des Controllers und legt es im Arbeitsspeicher ab

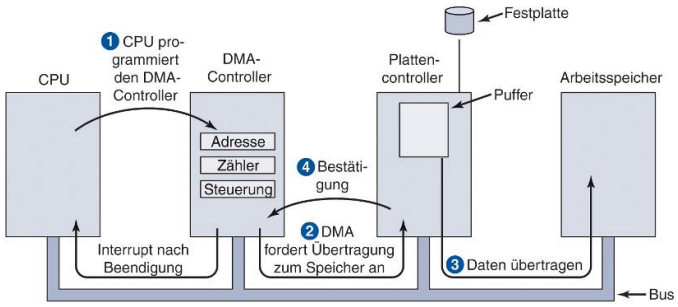
Hardware zur Ein- und Ausgabe

- Kommunikation Controller <--> Betriebssystem
  - **Direct Memory Access (DMA)**
    - **Lesen von Festplatte mit DMA**
      - Prozessor programmiert die Register des DMA-Controllers, damit er weiß, was er wohin transportieren soll (Schritt 1)
      - Kommando an Plattencontroller, damit dieser die Daten von der Platte in seinen internen Speicher einliest
      - Prüfsumme wird getestet
      - Sobald gültige Daten im Speicher des Plattencontrollers vorliegen, kann die Übertragung per DMA beginnen
      - DMA-Controller leitet den Datentransfer mit einem Lesebefehl über den Bus zum Plattencontroller ein (Schritt 2)
      - Speicheradresse des Ziels liegt auf den Adressleitungen des Busses, damit der Plattencontroller weiß, wohin er das nächste Zeichen aus dem Puffer schreiben soll



Hardware zur Ein- und Ausgabe

- Kommunikation Controller <--> Betriebssystem
  - **Direct Memory Access (DMA)**
    - **Lesen von Festplatte mit DMA (ff.)**
      - Schreiben in den Speicher ist Standard-Buszyklus (Schritt 3)
      - Sobald das Schreiben beendet ist, schickt der Plattencontroller ein Signal zur Bestätigung über den Bus zum DMA-controller (Schritt 4)
      - Der DMA-Controller erhöht die Speicheradresse und verringert die Anzahl der noch zu übertragenden Zeichen
      - Anzahl > 0, Schritt 2 bis 4 wiederholt
      - Anzahl = 0: Interrupt vom DMA-Controller
      - CPU weiß jetzt, daß Übertragung abgeschlossen
      - Wenn Betriebssystem die Arbeit wieder aufnimmt, ist der gewünschte Datenblock bereits im Speicher



Aufgabe/Frage

Wir kennen jetzt die Hardware und verschiedene Möglichkeiten die Controller der Hardware in das Betriebssystem einzubinden.

Zur Verwendung der Hardware muss auch Software bereitgestellt werden.

Frage:
Welches sind die Anforderungen an die Software?
Nennen Sie mindestens 2 Anforderungen. Bitte mit kurzer Erklärung.

Bedingung:
→ 5 min

Lösung

Antwort:
Software zur Ein- und Ausgabe

- **Geräteunabhängigkeit**
  - Bsp: Programm soll eine Datei einlesen, unabhängig davon ob diese auf Festplatte, CD-ROM oder USB-Stick liegt
- **Einheitliches Benennungsschema**
  - Der Name soll nicht von der Art des Gerätes abhängen
- **Fehlerbehandlung**
  - Fehler so nah wie möglich an der Hardware behandeln
    - evtl. durch Controller selbst
    - evtl. durch wiederholtes Lesen
- **Gleichheit unterbrechender (synchroner) und blockierender (asynchroner) Aufrufe**
  - Das Betriebssystem soll unterbrechende Aufrufe gegenüber dem Benutzerprogramm wie blockierende Aufrufe aussehen lassen
- **Puffermöglichkeit**
  - Die Daten sollen zwischenge-speichert werden können, um weitere Analysen durchzuführen
- **Unterscheidung von gemeinsam und exklusiv genutzter Geräte**
  - Bsp: Festplatte kann von allen Benutzern genutzt werden, aber CD-Brenner sollte nur einem Benutzer zur Verfügung gestellt werden

Software zur Ein- und Ausgabe

- Betriebssystem verwaltet Ressourcen
  - E/A-Geräte sind auch Ressourcen
  - Betriebssystem steuert auch alle Ein- und Ausgabegeräte, z.B.
    - Weiterleitung von Kommandos an die E/A-Geräte
    - Abfangen von Unterbrechungen (Interrupts) der E/A-Geräte
    - Behandlung von Fehlern
  - Abstraktion ist wesentliche Funktion des Betriebssystems
    - Programmierbarkeit von E/A-Geräten soll vereinfacht werden
    - **Geräteunabhängigkeit soll gewährleistet werden**
      Programme sollten so geschrieben werden können, daß sie auf jedes Ein-/Ausgabegerät zugreifen können, ohne das Gerät vorher spezifizieren zu

Software zur Ein- und Ausgabe

- Es gibt verschiedene Ansätze zur Durchführung von E/A
  - **Programmierte E/A**
    - Prozesssor hat die gesamte Arbeit
    - Der Auftrag wird an Controller geschickt und aktiv auf das Ergebnis gewartet (busy-wait)
      - ➢ Ausgabe auf einem Drucker

```
copy_from_user(buffer, p, count);          /* p ist der Kern-Puffer */
for (i = 0; i < count; i++) {              /* Schleife über alle Zeichen */
   while (*printer_status_reg != READY) ; /* Wiederhole bis zum Ende */
   *printer_data_register = p[i];          /* Gebe ein Zeichen aus*/
}
return_to_user( );
```

    - Nachteil: Prozessor ist komplett belegt
      - ➢ Aktives Warten (busy-wait)

Software zur Ein- und Ausgabe

- Ansätze zur Durchführung von E/A
  - **Interrupt-gesteuerte E/A**
    - Der Prozess beauftragt den Controller und kehrt sofort zurück
      - ➢ Auftraggebender Prozess wird ggfs. blockiert
    - Wenn der Auftrag erledigt ist, sendet der Controller einen Interrupt
      - ➢ Gerät ist nun wieder bereit
    - Interrupt wird behandelt
      - ➢ Auftraggebender Prozess wird ggfs. wieder auf Running gesetzt
    - Sinnvoll bei langsamen E/A-Geräten

Software zur Ein- und Ausgabe

- Ansätze zur Durchführung von E/A
  - **Interrupt-gesteuerte E/A**
    - Beispiel: Ausgabe auf Drucker
      - Systemaufruf für die Ausgabe der Zeichenkette
      - Puffer wird in den Kernadressraum kopiert
      - Erste Zeichen wird zum Drucker geschickt, sobald dieser bereit ist
      - Aufruf Scheduler, Scheduler teilt CPU anderen Prozess zu
      - Prozess, der das Drucken der Zeichenkette beauftragt, wird solange blockiert, bis die gesamte Ausgabe beendet ist (a)

Ausführung beim Systemaufruf zum Drucken

```
copy_from_user{buffer, p, count};
enable_interrupts();

while (*printer_status_read_reg
            != READY) ;
// ein Zeichen ausgeben
*printer_data_register = p[0];
// aktuellen Prozess blockieren
scheduler() ;
```

(a)

Software zur Ein- und Ausgabe

- Ansätze zur Durchführung von E/A
  - **Interrupt-gesteuerte E/A**
    - Beispiel: Ausgabe auf Drucker
      - Wenn Drucker mit dem Drucken des Zeichens fertig ist und für das nächste Zeichen bereit ist: Interrupt
      - Interrupt hält den gesamten Prozess an und speichert dessen Zustand
      - Ausführung Unterbrechungsroutine für den Drucker (b)
      - Keine Zeichen mehr zum Drucken: blockierte Benutzerprozess kann weiterabreiten
      - Andernfalls: nächstes Zeichen wird gedruckt, der Interrupt bestätigt und der Prozess läuft an derselben Stelle weiter, an der er unterbrochen wurde.

Nachteil: Interrupterzeugung kostet Zeit

Ausführung bei Unterbrechung (Interrupt-Handler)

```
if (count==0 ) {
  unblock_user();
}
else {
  // ein Zeichen ausgeben
  *printer_data_register = p[i];
  count = count -1 ;
  i=i+1;
}
// interrupt bestätigen
acknowledge_interrupt();

return_from_interrupt();
```

(b)

Software zur Ein- und Ausgabe

- Ansätze zur Durchführung von E/A
  - **DMA-basierte E/A**
    - DMA-Controller koordiniert die Datenübertragung
      - Ohne CPU zu belasten
    - DMA-Controller schreibt nacheinander die Zeichen in das Datenregister des Druckers, ohne Prozessorbeteiligung
    - Die Datenübertragung wird durch Controller gestartet
      - Parameter sind Geräte-Adresse, Startadresse, Länge der Daten und Transferrichtung an DMA-Controller
      - Nach Beendigung wird vom DMA-Controller ein Interrupt gesendet
    - Vorteile:
      - Verringerung der Interrupts
      - Entlastung der CPU
      - Sinnvoll bei Übertragung größerer Datenblöcke, weil DMA-Controller oft langsamer als CPU

Software zur Ein- und Ausgabe

- Ansätze zur Durchführung von E/A
  - **DMA-basierte E/A**
    - Beispiel: Ausgabe auf Drucker (vgl. Interrupt-gesteuerte E/A)

Ausführung beim Systemaufruf zum Drucken

```
copy_from_user{buffer,  p,  count};

setup_DMA_controller();

// aktuellen Prozess blockieren
scheduler() ;
```

Ausführung bei Unterbrechung nach Datenübertragung

```
acknowledge_interrupt();

unblock_user();

return_from_interrupt();
```

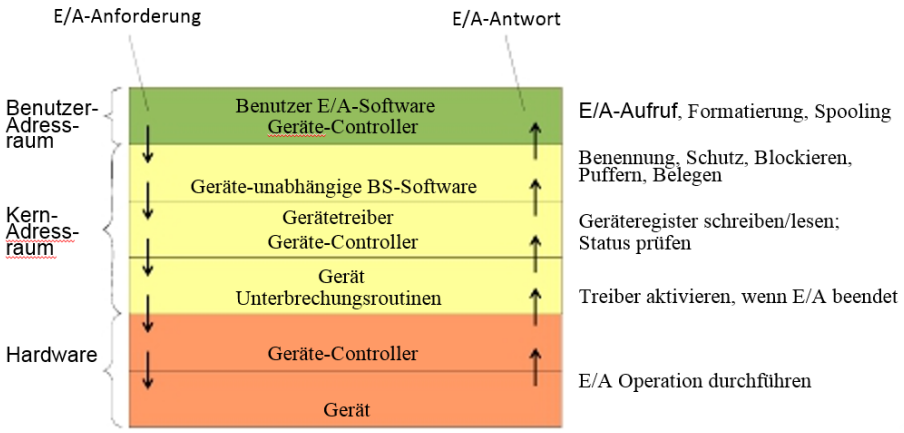
Software zur Ein- und Ausgabe

- **Allen Ansätzen gemeinsam**
  - Funktionen sind als Systemaufrufe an die Geräte realisiert
    - Oft mit Unterbrechungsroutinen

- Wie schaut eine E/A-Softwarearchitektur aus?

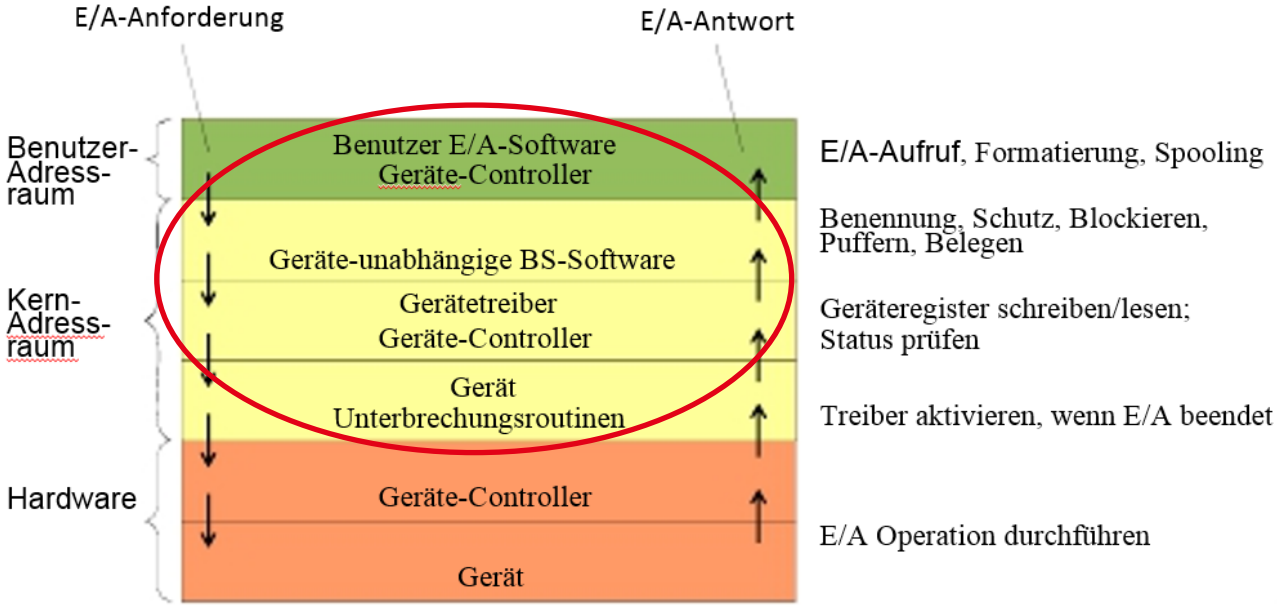
Software zur Ein- und Ausgabe

- Schichten-Architektur der E/A-Software
  - 4 Schichten (Unterbrechungsroutinen, Gerätetreiber, geräteun-abhängige Ein-/Ausgabesoftware, Ein-/Ausgabesoftware im Benutzeradressraum)
  - Bsp. Kontrollfluss
    Ein Benutzerprogramm will einen Block aus einer Datei lesen
  - Betriebssystem wird mit der Ausführung beauftragt
  - Die geräteunabhängige Software überprüft den Cache, ob der Block enthalten ist
  - Falls nicht, wird der Treiber beauftragt, die Anfrage an die Hardware weiterzuleiten, die den Plattenzugriff ausführt.
  - Der Prozess wird dann so lange blockiert, bis die Plattenoperation beendet ist und die Daten im Puffer des Aufrufers stehen.
  - Sind die Daten im Puffer , erzeugt die Hardware einen Interrupt.
  - Die Unterbrechungsroutine stellt fest, welches Geräte den Interrupt ausgelöst hat.
  - Der Status dieses Geräts wird entnommen und der schlafende Prozess geweckt.
  - Geweckter Prozess beendet seine Ein-/Ausgabeanfrage.
  - Benutzerprozess kann weiterlaufen



Software zur Ein- und Ausgabe

- Schichten-Architektur der E/A-Software
  - 4 Schichten (Unterbrechungsroutinen, Gerätetreiber, geräteunabhängige Ein-/Ausgabesoftware, Ein-/Ausgabesoftware im Benutzeradressraum)
  - Jede Schicht hat eine genau festgelegte Aufgabe und
  - eine wohldefinierte Schnittstelle zu den angrenzenden Schichten



Software zur Ein- und Ausgabe

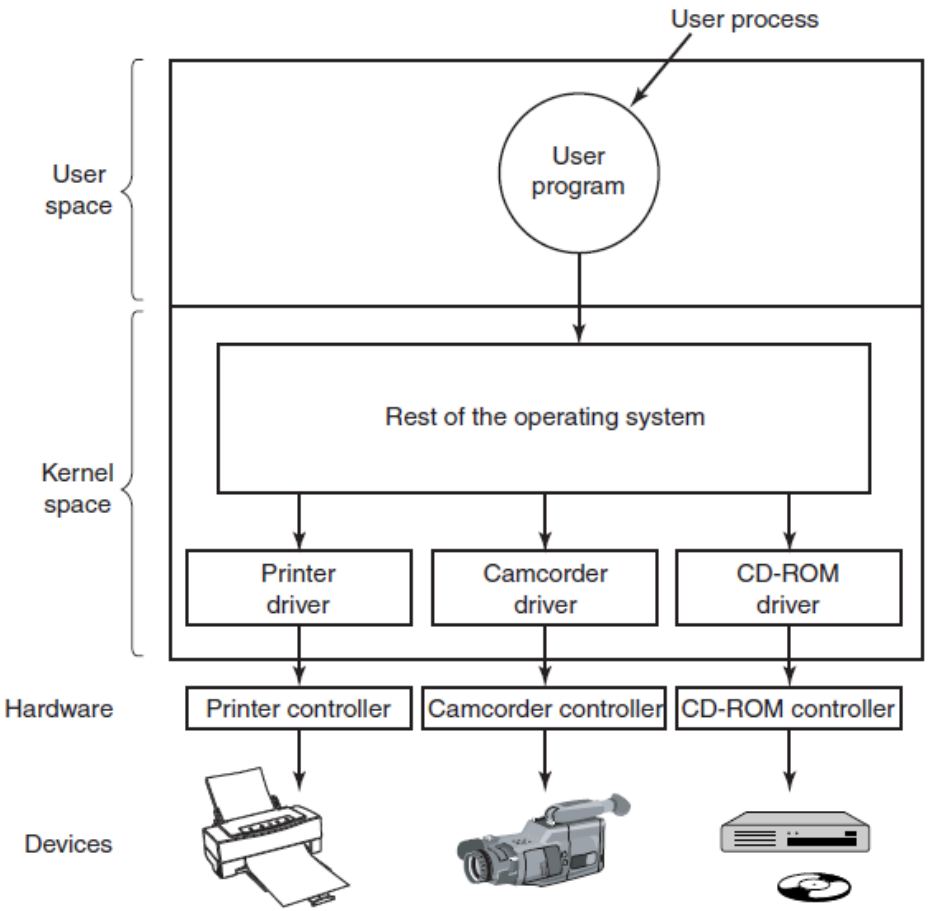
- Schichten-Architektur der E/A-Software
  - **Unterbrechungsroutinen**
    - Für die meisten Ein-/Ausgaben gehören Interrupts zum unangenehmen Teil der Realität. Sie sollten tief im Innern des Betriebssystems verborgen werden.
    - Interrupts zu verbergen, ist den Treiber, der die Ein-/Ausgabeoperation gestartet hat, zu blockieren, bis die Operation beendet wird.
    - Treiber kann sich auch selbst blockieren, in dem er beispielsweise ein *down* auf einen Semaphor macht, ein *wait* auf einer Zustandsvariablen oder ein *receive* bzgl. einer Nachricht ausführt.
    - Tritt der Interrupt auf, muss Unterbrechungsroutine den Interrupt behandeln
    - Danach wird blockierter Treiber freigegeben, durch
      - Up-Operation auf einen Semaphor
      - ‚signal' auf eine Zustandsvariable im Monitor oder
      - durch das Senden einer Nachricht an den blockierten Treiber

Software zur Ein- und Ausgabe

- Schichten-Architektur der E/A-Software
  - **Gerätetreiber**
    - Jedes Ein-/Ausgabegerät, das an einen Computer angeschlossen ist, hat geräteabhängige Steuersoftware (Gerätetreiber, device driver)
    - Der Gerätetreiber wird vom Hersteller des Geräts codiert
      - Beispiel:
        - SCSI-Treiber verwaltet meist mehrere SCSI-Platten verschiedener Größen und unterschiedlciher Geschwindigkeiten
        - Maus und Joystick sind so unterschiedlich, daher verschiedene Treiber

Software zur Ein- und Ausgabe

- Schichten-Architektur der E/A-Software



Logische Positionierung der Gerätetreiber.
In Realität erfolgt jede Kommunikation zwischen Treibern und den Controllern über den Bus.

Aufgabe/Frage

Wir hatten gerade gesagt, daß jedes Ein-/Ausgabegerät, das an einen Computer angeschlossen ist, geräteabhängige Steuersoftware (Gerätetreiber) hat.

Frage:
Kennen Sie eine universelle Bustechnologie, an die sehr viele Devices angeschlossen werden können?

Wie funktioniert das Ihrer Meinung nach?

Bedingung:
→ ad hoc

Lösung

Antwort:

- USB ist eine serielle Bustechnologie, die universal ist
- An die USB-Schnittstelle lassen sich Festplatten, Speichersticks, Kameras, Mäuse, Tastaturen, Roboter, Lesekarte für Kreditkarten, Discokugeln und Kaffeetassenwärmer anschliessen.
- USB ist in Schichten aufgebaut
  - drei Schichten Physical Layer, Link Layer und Protocol Layer
  - Physical Layer für die Verbindung zwischen Host und Gerät bzw. zwischen Hub und Gerät.
  - Der Link Layer (»Sicherungsschicht«) hält die Verbindung aufrecht und sorgt durch Fehlererkennung für die Integrität der übertragenen Daten. Auf dieser Ebene werden die Pakete erstellt und Link-Befehle ausgegeben.
  - Die Protokollschicht schließlich verwaltet den Datenverkehr zwischen Gerät und Host.
  - Speichermedien, die mit einem USB-Anschluss ausgestattet sind, können im laufenden Betrieb mit einem Computer verbunden werden.
  - USB 3.0 kann maximal 4 GBit/s übertragen (USB 2.0: 480 MBits/s). Mit einem USB-3.0-Controller und ein USB-3.0-Kabel, kann der Datentransfer mit maximal 5 GBit/s (Super-Speed-Modus) stattfinden. Gegenüber der Vorgänger-Technologie USB 2.0 ist diese neue Variante mit fünf zusätzlichen Kontakten ausgestattet

Zusammenfassung

- E/A-Hardware: Controller
- Kommunikation Controller und CPU
  - Direct Memory Access (DMA)
- Anforderungen an E/A-Software
  - Geräteunabhängigkeit
- Durchführung der E/A
  - Programmierte E/A, Interrupts, DMA
- Schichtenmodell der E/A-Software

Hardware von Plattenspeichern

- Vergleich der Parameter eines Standardspeichermediums des originalen IBM-PC mit den Parametern einer Festplatte, die 40 Jahre später hergestellt wurde

Parameter	360-KB-Diskette von IBM	WD-3000-HLFS
Anzahl der Zylinder	40	36481
Spuren pro Zylinder	2	255
Sektoren pro Spur	9	63
Sektoren pro Platte	720	586072368
Bytes pro Sektor	512	512
Plattenkapazität	360 KB	300GB
Zugriffszeit (benachbarter Zylinder)	6 ms	0,7 ms
Zugriffszeit (Durchschnitt)	77 ms	4,2 ms
Rotationszeit	200 ms	6 ms
Motoranlauf- und -auslaufzeit	250 ms	
Übertragungszeit eines Sektors	22 ms	1,4 micro sec

Abbildung 5.18: Plattenparameter der ursprünglichen 360-KB-Diskette des IBM-PCs und einer WD-18300-Festplatte von Western Digital (Forts.)

Hardware von Plattenspeichern

- Geometrie, die von der Treibersoftware verwendet wird, weicht vom physischen Format ab
- Früher : Anzahl der Sektoren pro Spur für alle Zylinder gleich
- Heute: Einteilung in Zonen, wobei die äusseren Zonen mehr Sektoren als die Inneren haben
- Anzahl der Sektoren pro Spur wächst mit jeder Zone von innen nach außen (ca. 4%)

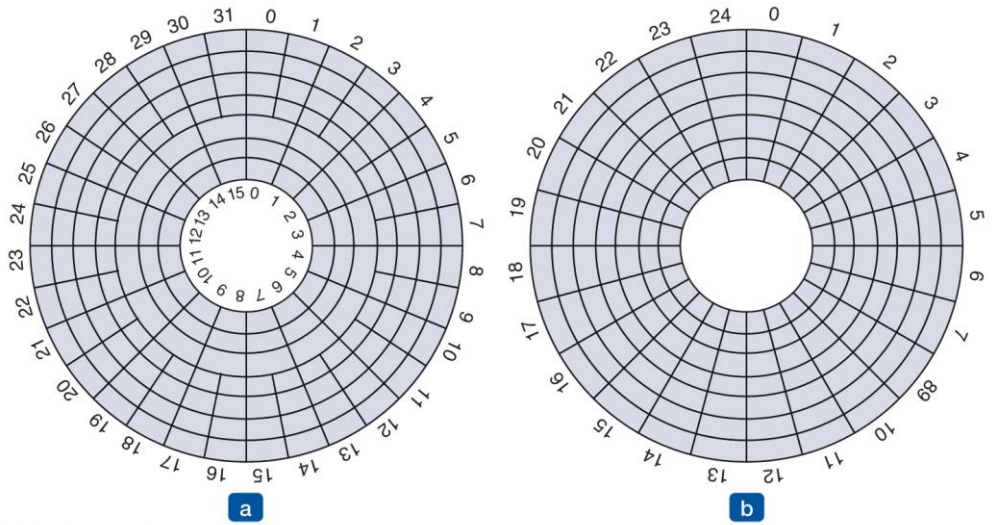


Abbildung 5.19: (a) Physische Geometrie einer Platte mit zwei Zonen (b) Eine mögliche virtuelle Geometrie für diese Platte

(a)

- Kleine Platte mit 2 Zonen
- Äußere Zone besitzt 32 Sektoren pro Spur
- Innere Zone besitzt 16 Sektoren pro Spur

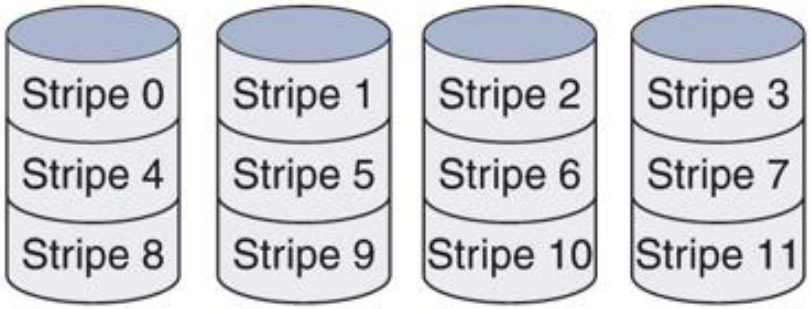
- Beide Platten haben 192 Sektoren,
- nach außen gezeigte Anordnung ist anders als die reale Anordnung
- Eine reale Platte hat typischerweise 16 oder mehr Zonen

RAID (Redundant Array of Inexpensive Disks)
RAID (Redundant Array of Independant Disks)

- Prozessorgeschwindigkeit wächst exponentiell (alle 18 Monate verdoppelt)
- Festplattengeschwindigkeit ist gering gestiegen (ca. 50 bis 100 ms)
- Lösung: Parallelverarbeitung (nicht nur bei CPU)
- Vorschlag von Patterson et al. (1988) sechs verschiedene Plattenorganisationen
- Mit RAID Zusammenschaltung mehrerer physikalischer Festplatten zu einem großen logischen Laufwerk
- Definition der verschiedenen Möglichkeiten in RAID-Leveln
- Offiziell: 8 RAID-Level (0 bis 7), nur Level 0 bis 5 sind spezifiziert.
- Durchgesetzt haben sich: RAID-Level 0, 1 und 5
- RAID-Level 2, 3 und 4 spielen in der Praxis keine Rolle.

RAID 0

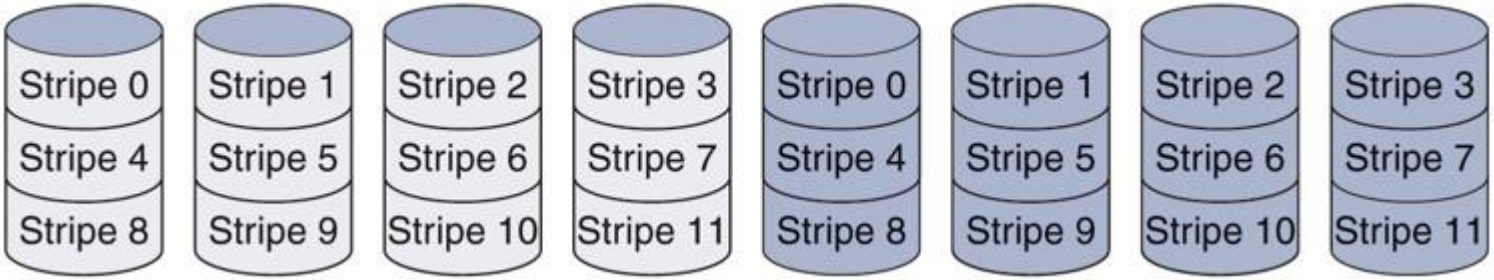
- RAID0 ist eigentlich kein RAID; es ist wohl ein Array, aber nicht redundant.
- RAID0 meint **Striping**: Verteilung der Daten gleichmäßig auf alle Platten. Es entsteht sehr große virtuelle Festplatte.
- Sektoren 0 bis k-1 sind Stripe 0, k bis 2k-1 sind Stripe 1, usw.
- Bei RAID-Level 0: Schreiben der aufeinanderfolgenden Stripes nach Round Robin
- Striping bedeutet vor allem Performancegewinn
- Sicherheit: Geringer als bei einzelnen Festplatten → Ausfall einer Platte bedeutet Verlust aller Daten.



RAID 0 mit 4 Festplatten, z.B. jede Platte hat mittlere Lebensdauer von 20.000 h
→ alle 5.000 h fällt eine Festplatte aus
→ alle Daten verloren

RAID 1

- RAID1 bedeutet **Spiegelung**
- RAID 1 verdoppelt die Sicherheit
- Schreiben der Daten ohne Unterschied auf beiden Platten
- Schreibzugriff: genauso schnell wie bei einer einzelnen Platte
- Lesezugriff: Controller liest die eine Hälfte der Daten von der einen Platte, die andere Hälfte von der anderen.
- Fehlertoleranz ist ausgezeichnet, bei Ausfall einer Platte → Kopie vorhanden
- Kaputte Platte wird ersetzt; Danach Kopieren der Daten von Sicherungs-laufwerk auf neu ersetzte Platte
- Verdopplung der Kosten



RAID 1

RAID 2

- RAID0 und RAID1: Stripes von Sektoren
- RAID2: auf Wortbasis, evtl. auf Byte-Basis
- Armpositionierung und Plattenrotation sollten synchron arbeiten

RAID 2

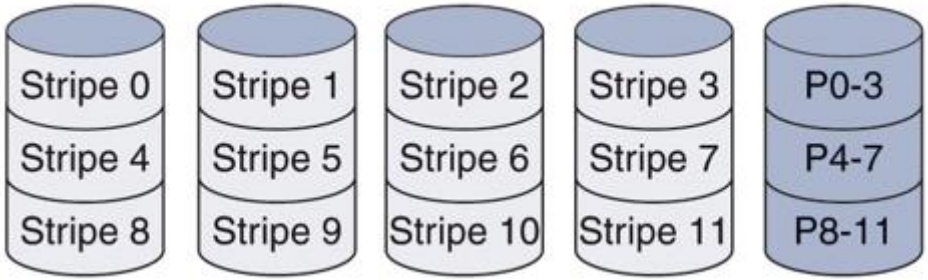
RAID 3

- RAID3: vereinfachte Version von RAID2
- Berechnung eines einzelnen Paritätsbit für jedes Datenwort und Schreiben auf Paritätsplatte
- Exakte Synchronisierung der Festplatten (wie bei RAID2)
- RAID3: Verwendung einer einzigen Parity-Platte → Senkung der Kosten der Redundanz gegenüber der Spiegelung
- Die Parity-Platte sorgt dafür, dass die Quersumme aller Platten für jedes einzelne Bit gerade bleibt
- Die Daten werden auf die restlichen Platten geschrieben (Striping)
- Fällt eine Platte aus, kann sie aus allen anderen neu berechnet werden

RAID 3

RAID 4

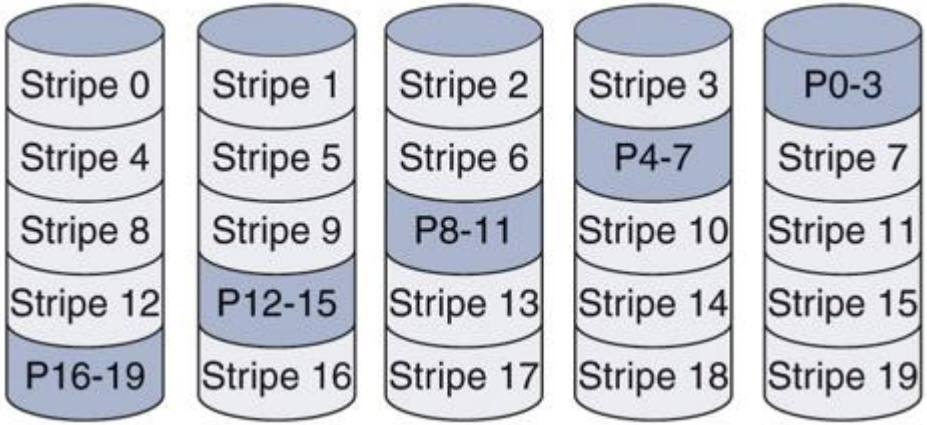
- RAID4 arbeitet wieder mit Stripes anstatt mit einzelnen Wörtern und Paritäten
- Keine synchronisierten Laufwerke notwendig
- Entspricht RAID0, allerdings mit einer Stripe-zu-Stripe-Parität, die auf eine Paritätsplatte geschrieben wird
- Wenn eine Platte zerstört wird, können die verlorenen Bytes von der Paritätsplatte wieder berechnet werden, in dem alle Platten gelesen werden
- Geschwindigkeit bei kleinen Änderungen ist schlecht
- Paritätsplatte kann zum Engpass werden



RAID 4

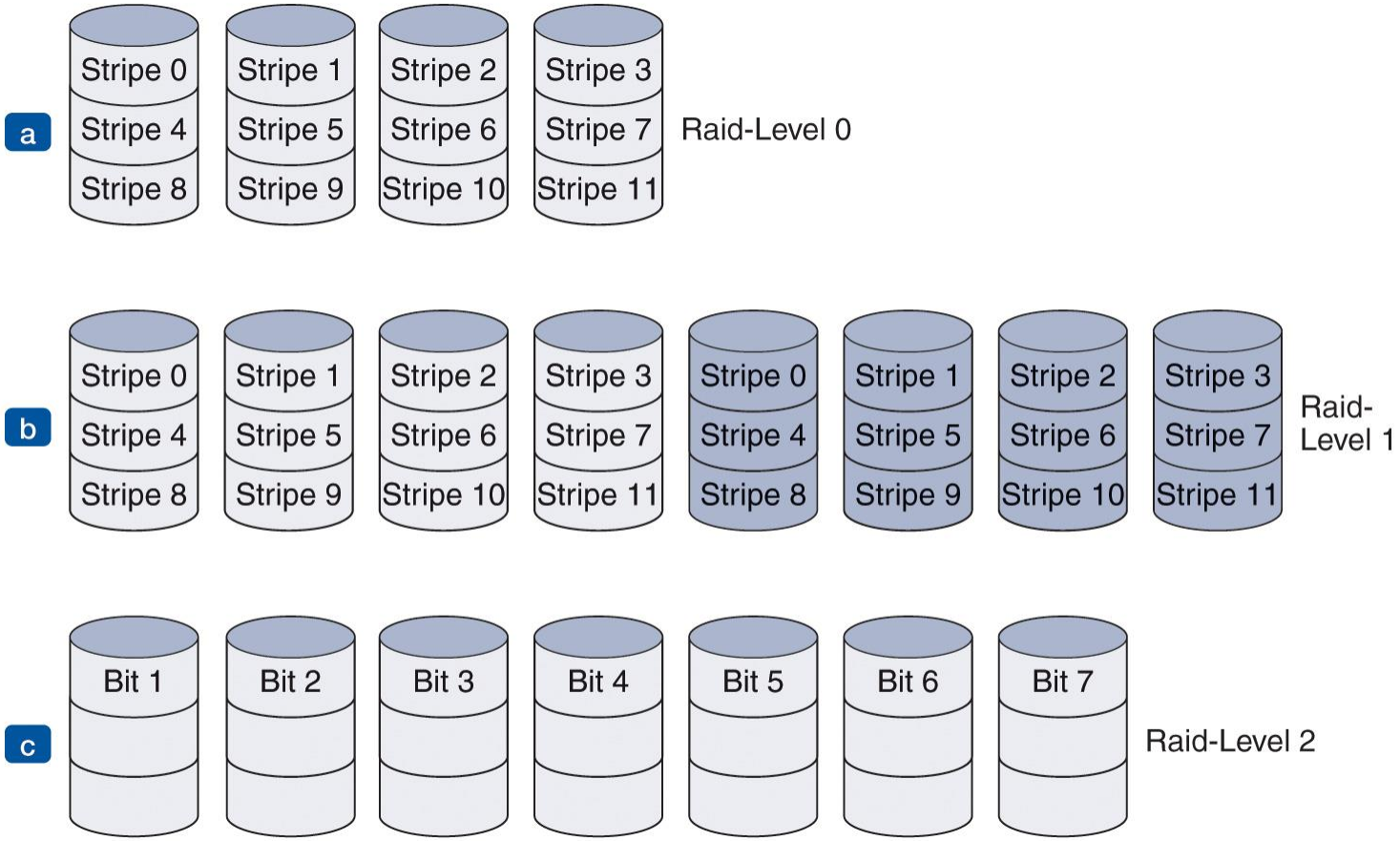
RAID 5

- RAID5 arbeitet wie RAID4
- Keine Synchronisierten Laufwerke notwendig
- RAID5 schreibt die Parity-Informationen gleichmäßig im Round Robin-Verfahren über alle Festplatten
- RAID5 ist dadurch beim Schreiben schneller, indem es den Flaschenhals beim Schreiben der Parity auf eine einzige Platte vermeidet
- Wiederherstellung nach einem Plattenausfall ist aufwendig



RAID 4

RAID Zusammenfassung



RAID Zusammenfassung

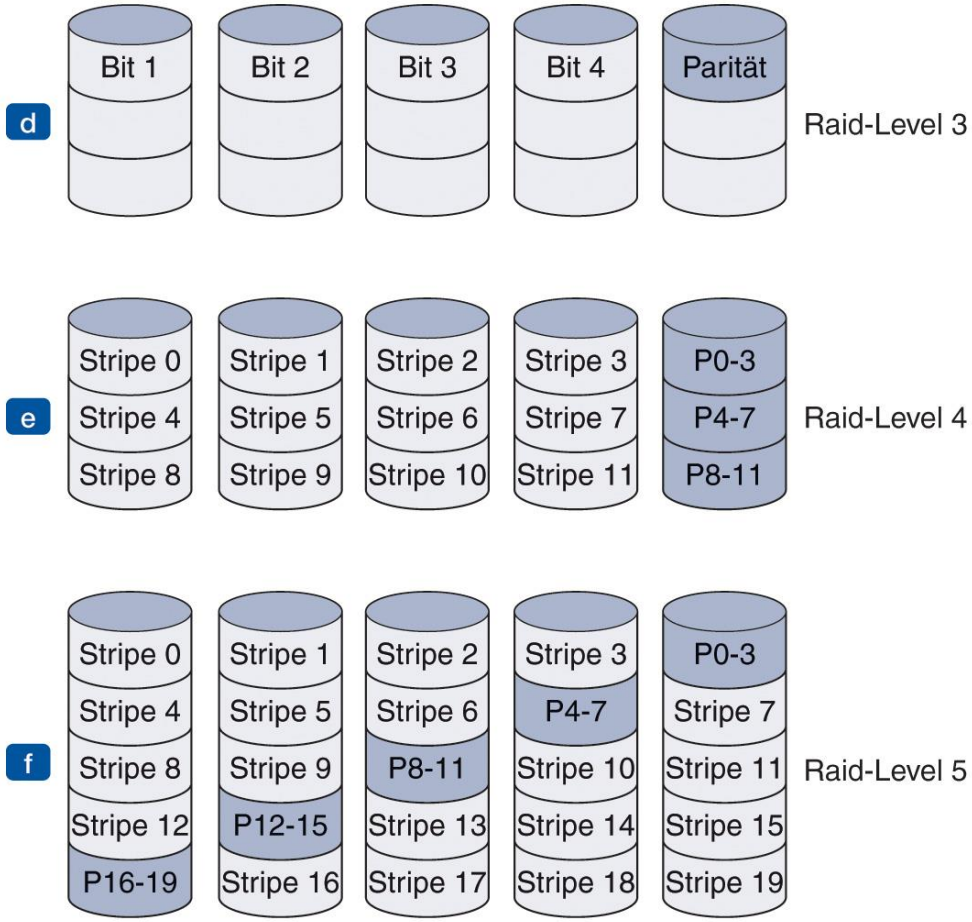


Abbildung 5.20: RAID-Level 0 bis 5. Die Sicherungs- und Paritätslaufwerke sind dunkler dargestellt.

RAID Zusammenfassung

RAID-Level	Lesegeschwindigkeit	Schreibgeschwindigkeit	Datensicherheit	Speicherkapazität
RAID 0	+ +	+ +	- -	+
RAID 1	+ +	+	+ +	- -
RAID 5	+	-	+	-

Bewertungen sind ein Vergleich zu einer einzelnen Platte (neutral oder Null).

- RAID 0: Nicht ausfallsicher, dafür schnelle Lese- und Schreibgeschwindigkeit
- RAID 1: Ausfallsicher, aber teuer
- RAID 5: Ausfallsicher, je nach Controller langsame Schreibgeschwindigkeit

RAID-Level im Vergleich

Betrieb	RAID 0	RAID 1	RAID 5	RAID 6
Redundanz	nein	ja	ja	ja
min. Datenträger	2	2	3	4
Rechenaufwand	sehr gering	sehr gering	mittel (XOR)	hoch
Datentransferrate	höher als Einzelplatte	beim Lesen höher als Einzelplatte	*abhängig vom Controller	*abhängig vom Controller
Kapazität bei 2 Platten	2	1	nicht möglich	nicht möglich
Kapazität bei 3 Platten	3	nicht möglich	2	nicht möglich
Kapazität bei 4 Platten	4	2	3	2
Kapazität bei 5 Platten	5	nicht möglich	4	3

Tabelle: Elektronik-Kompendium, https://www.elektronik-kompendium.de/sites/com/1001011.htm

Aufgabe/Frage

Allgemeine Fragen:

- Auf was zielt RAID ab?

- Welche Arten von RAID gibt es? Kennen Sie noch die Namen?

- Warum ist RAID 0 eigentlich kein RAID-Level?
  (Hinweis: es fehlt etwas Wichtiges!)

Bedingung:

→ 5 min

Lösung

Antwort

- **Auf was zielt RAID ab**?
  - Daten sicher speichern, ohne dass diese verloren gehen. Meist redundantes Verfahren.

- **Welche Arten von RAID gibt es?**
  - RAID 0 – Data-Striping
  - RAID 1 – Mirroring und Duplexing. ...
  - RAID 1E, RAID 1E0. ...
  - RAID 2 - Hamming Code ECC. ...
  - RAID 3 – Data-Striping plus Parity. ...
  - RAID 4 – Data plus Parity. ...
  - RAID 5 – Data- and Parity-Striping. ...
  - RAID 6/RAID DP – Double-Parity.

- **Warum ist RAID 0 eigentlich kein RAID-Level?**
  - es fehlt die Redundanz

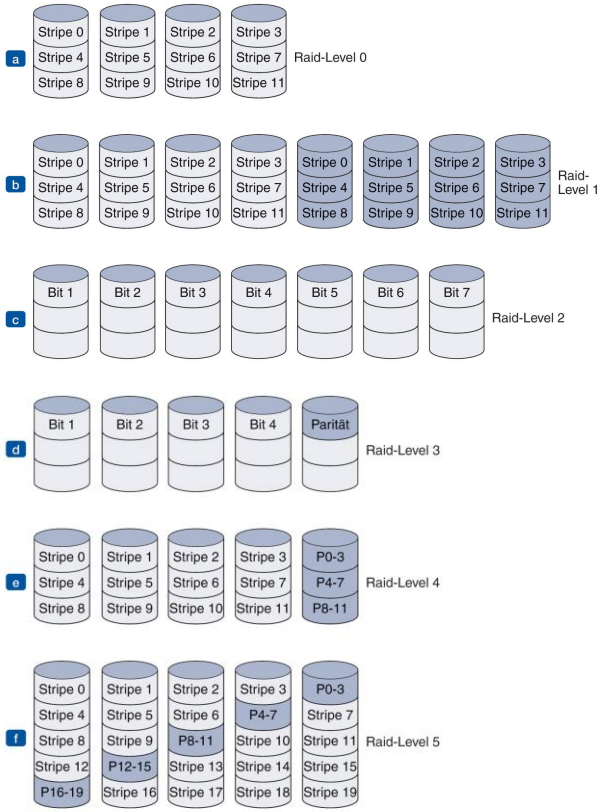


Abbildung 5.20: RAID-Level 0 bis 5. Die Sicherungs- und Paritätslaufwerke sind dunkler dargestellt.

Aufgabe/Frage

Wir haben jetzt einige RAID-Level kennengelernt.

Frage:
Was versteht man unter dem RAID-Level 15?

Wieviel Festplatten sind notwendig?

Bedingung:
→ 5 min

Lösung

Antwort

- **Was versteht man unter dem RAID-Level 15?**
  - RAID 15 ist ein Kombination aus RAID 1 und RAID 5.
    RAID 5= Daten werden über mehrere Festplatten verteilt (Stripe Set) und mit Paritätsbits abgespeichert. Es wird Sorge getragen, daß ein Laufwerk nicht seine eigenen Paritätsbits speichert, sondern auf den anderen Festplatten hinterlegt werden.
    Dieses Verfahren wir gekoppelt mit einer Spiegelung der Festplatten (RAID1).

- **Wieviel Festplatten sind notwendig?**
  - RAID5 = mindestens 3 Festplatten PLUS weitere 3 Festplatten für Spiegelung

Ein- und Ausgabe

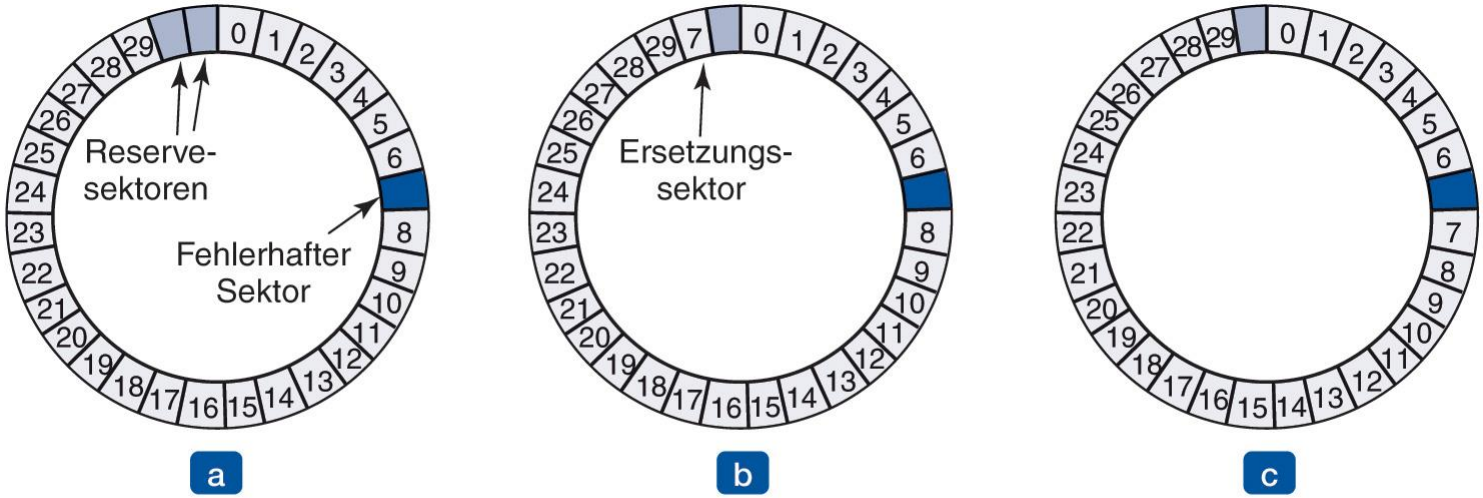


Abbildung 5.30: (a) Eine Plattenspur mit einem fehlerhaften Sektor (b) Ersetzung des defekten Sektors durch einen Reservesektor (c) Verschiebung des Sektors, um den fehlerhaften Sektor zu umgehen

Zusammenfassung

- RAID: (Redundant Array of Inexpensive Disks)
  RAID (Redundant Array of Independant Disks)

- Zusammenschaltung mehrerer physikalischer Festplatten zu einem großen logischen Laufwerk

- Definition der verschiedenen Möglichkeiten in RAID-Leveln

- Durchgesetzt haben sich: RAID-Level 0, 1 und 5